Departamento de Informática

PROYECTO FIN DE CARRERA

CREACIÓN DE UN PORTAL DE SINDICACIÓN DE NOTICIAS

Autor: Miguel Vidal Hernando

Tutor: David Palomar Delgado

Colmenarejo, Junio 2015

Título: Creación de un portal de sindicación de noticias.
Autor: Miguel Vidal Hernando
Director: David Palomar Delgado

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 19 de Junio de 2015 en Colmenarejo, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Quiero dar mi agradecimiento a David Palomar por haberme dirigido durante la realización de este proyecto, por sus consejos y orientación y por tener una paciencia infinita.

Agradecer a mis padres por darme la oportunidad de llegar donde he llegado y animarme a continuar mejorando.

Agradecer a mi hermano por su apoyo continuo, tanto en los momentos buenos como en los no tan buenos.

Por último agradecer a mis compañeros de universidad el gran ambiente creado durante estos años que me ha empujado hasta el final.

Resumen

El proyecto consiste en la creación de un portal web de sindicación de noticias, concretamente información deportiva. El portal obtiene información actualizada de diferentes fuentes web y la ofrece para que los usuarios del mismo puedan acceder a ella de forma rápida y centralizada, evitando de este modo tener que visitar cada uno de los diversos sitios web de información deportiva.

El portal se ha desarrollado usando componentes portlets, que son aplicaciones web independientes que se ven como pequeñas ventanas dentro del portal que los aloja.

Para obtener la información a mostrar se han utilizando diferentes tecnologías que se ofrecen en la actualidad en la red. Concretamente se han utilizado tres tecnologías diferentes: RSS, servicios web basados en SOAP y servicios web basados en REST. Usando cada una de ellas se han creado diferentes portlets que ofrecen información sobre temas deportivos diversos y que aunque son capaces de funcionar por separado, en conjunto conforman el portal.

Palabras clave: Portal; Sindicación; Fuentes web; Centralización, Portlets, RSS, Servicios web basados en SOAP; Servicios web basados en REST.

Abstract

The project consist of the creation of a web portal for news syndication, more specifically sports information. The information is obtained from different web sources and is offered on the portal so that users can access it quickly and centrally, avoiding visiting each of the diverse sports information websites.

The portal has been developed using portlet components, which are independent web applications that look like small windows within the portal that hosts them.

In order to obtain the information to be displayed, different types of technology which are available nowadays on the web have been used. Specifically three different types of technology have been used: RSS, SOAP-based web services and REST-based web services. Using each one of these several portlets have been created and although they are able to operate separately, all the portlets together make up the application in its entirely.

Keywords: Portal, Sydication, Web resources, Centralisation, Portlets, RSS, SOAP-based web services, REST-based web services.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Introducción

Este primer capítulo de la memoria hace un resumen general de toda la documentación recogida en ella. Comienza describiendo en qué consiste la aplicación creada y cuáles son los objetivos a conseguir con la realización de este proyecto. También resume las fases llevadas a cabo durante su desarrollo, definiendo su duración y en que han consistido. Se exponen los medios empleados para la realización de la aplicación, tanto hardware como software y finalmente se describe la estructura que sigue la memoria para que resulte fácil de seguir.

1.2 Descripción del proyecto

El proyecto consiste en la creación de un portal web de sindicación de información deportiva. Un sindicador de contenidos es un tipo de software, que puede ser de escritorio o web, el cual recoge información de diferentes fuentes web y la dispone de una forma en que los usuarios pueden acceder a ella de una manera centralizada y rápida sin tener que visitar numerosos sitios web diferentes. La información que se ofrece estará en todo momento actualizada ya que son los propios sitios web los que añaden la nueva

información y es el sindicador el encargado de comprobar las actualizaciones y mostrarlas de forma centralizada.

A la hora de desarrollar el portal, se pretende que este se conforme de componentes modulares. Esto quiere decir que los distintos contenidos se ofrecerán en pequeñas ventanas como si fueran pequeñas aplicaciones independientes. Estos componentes se llaman portlets. El portal reúne una serie de portlets, que conjuntamente forman el portal de noticias completo. La ventaja de ofrecer la información en portlets en vez de cómo lo hace un diario deportivo online normal, es que estos portlets permiten al usuario que elija la información que le interesa y pueda eliminar los portlets con información en la que no esté interesado. Además, estos portlets se pueden mover y colocar en la posición que se desee dentro del portal por lo que también se le ofrece la posibilidad de que el usuario cree su propia maquetación. En resumen, trabajar con portlets le ofrece al usuario una gran personalización, que es una de las características principales de los portales de sindicación de contenidos.

Cada uno de estos portlets recoge información utilizando diferentes tecnologías que se ofrecen en la actualidad en la red. Concretamente se utilizan tres tecnologías diferentes: RSS, servicios web basados en SOAP y servicios web basados en REST. Usando cada una de ellas se han creado diferentes portlets que ofrecen información deportiva sobre diferentes temas.

Utilizando la tecnología RSS, se ha creado un lector RSS personalizado en forma de slideshow. Este ofrece diferentes fuentes RSS de los principales medios españoles de información deportiva para que el usuario pueda elegir cuál de ellos leer en cada momento. Además se le ofrece la posibilidad de acceder a la noticia original con la información completa cuando se hace clic sobre uno de los titulares mostrados.

Utilizando servicios web basados en REST se creará un portlet lector de tweets de los equipos de primera división y de algunos medios importantes de información deportiva. Este portlet ofrecerá la posibilidad de leer los últimos tweets escritos por los usuarios seguidos, pero también de leer otros tweets más antiguos mediante un selector de página.

Con el objetivo de aprender la tecnología SOAP, se ha creado un servicio web proveedor de información que ofrece información de las estadísticas sobre la liga de fútbol de primera división. Utilizando este servicio web se han creado una serie de portlets clientes que consumen la información ofrecida por este. Concretamente se ha desarrollado un portlet con la clasificación de la liga, otro con las estadísticas de goles asistencias y amonestaciones y otro con la información de las plantillas de cada equipo.

También se han creado varios portlets con información sobre el Mundial de Brasil de 2014, que aunque se celebró el año pasado, es uno de los pocos servicios web SOAP de información deportiva que se ofrecen en la red. Concretamente se ha desarrollado un portlet que ofrece la clasificación de los goleadores del mundial y otro con información sobre las plantillas de las diferentes selecciones e información de los estadios donde se jugaron los partidos.

Debido a que los portlets pueden ser cambiados de posición y por lo tanto su tamaño puede variar, la maquetación de la información mostrada en cada uno de los portlets

puede no quedar optimizada dependiendo del tamaño de estos. Se ha tenido este problema en cuenta y se ha procurado que la maquetación se actualice en tiempo de ejecución para ajustarse al tamaño variable de los portlets.

Tras el resumen general sobre lo que va el proyecto se van a exponer los objetivos principales de este:

- Creación de un portal de sindicación de información deportiva.
  - Crear portlet que muestre información basado en servicios web SOAP.
  - Crear portlet que muestre información basado en servicios web REST.
  - Crear portlet que muestre información basado en RSS
  - Actualización automática del contenido.
- Desarrollar el contenido como componentes modulares.(Portlets)
  - Ofrecer personalización al usuario.
  - Optimizar maquetación en tiempo de ejecución.

1.3 Objetivos personales

Además de los objetivos del propio proyecto me he planteado unos objetivos personales que no solo me valdrán pasa sacar este proyecto adelante, sino que me podrían ayudar en mi futura carrera profesional.

- Estudio y aprendizaje del lenguaje de programación Java.
- Estudio y aprendizaje sobre Liferay.
- Estudio y aprendizaje sobre el funcionamiento de los portlets.
- Estudio y aprendizaje del funcionamiento de los servicios web basados en SOAP, tanto de los proveedores de servicio como los clientes consumidores.
- Estudio y aprendizaje del funcionamiento de los servicios web basados en REST.
- Estudio y aprendizaje del formato de datos RSS así como el funcionamiento de un lector RSS.
- Estudio y aprendizaje de la tecnología EJB.
- Estudio y aprendizaje de la tecnología AJAX.

1.4 Fases del desarrollo

A la hora de desarrollar el proyecto se han seguido las fases típicas en el desarrollo de un proyecto informático. Se comenzó con la fase de estudio, y le siguieron la fase de análisis, diseño, desarrollo y pruebas. Todo el proyecto tuvo una duración de 7 meses y medio o lo que es lo mismo 163 días laborales. A continuación se exponen la duración de cada una de ellas y cuál es su objetivo.

- Fase de reconocimiento del problema y estudio: Fue la fase de mayor duración de todo el proyecto debido a la gran cantidad de conceptos nuevos que tuve que aprender. Tuvo una duración de 260 horas repartidas en 65 días e incluyó el estudio del problema a resolver, buscando herramientas y tecnologías que me permitieran resolver el problema planteado, el aprendizaje del lenguaje de programación Java del que tenía muy pocos conocimientos, el aprendizaje y familiarización con las diferentes tecnologías y herramientas seleccionadas y la instalación del entorno de desarrollo, lo cual me resulto de gran dificultad hasta que conseguí su correcto funcionamiento debido a problemas de compatibilidad entre los diferentes componentes.
- Fase de análisis: Esta fase tuvo una duración de 48 horas repartidas en 12 días. En ella se recogieron los requisitos y restricciones que debía tener la aplicación. Durante esta fase se obtuvieron los documentos completos de casos de uso y de requisitos de software.
- Fase de diseño: En la fase de diseño se organizaron de forma adecuada los datos obtenidos en la fase de análisis para su posterior implementación. Tuvo una duración de 72 horas repartidas en 18 días. En esta fase de diseño se definieron la arquitectura del sistema, tanto física como lógica y se realizo el diseño estático y dinámico de cada uno de los componentes desarrollados. También se incluye el diseño de la base de datos utilizada.
- Fase de desarrollo: Fue la segunda fase más larga. Tuvo una duración de 140 horas repartidas en 35 días. Consiste en la codificación de lo planteado en la fase de diseño incluida la configuración del portal y la creación de los diferentes portlets planteados utilizando las tecnologías aprendidas.
- Fase de pruebas: La fase de pruebas fue la más corta de todo el proyecto con una duración de 40 horas repartidas en 10 días. En ella se realizaron las pruebas recogidas en el plan de pruebas. El objetivo de esta fase es asegurar que la aplicación hace lo que se planteó en la fase de análisis y encontrar posibles errores en la codificación.

Fase de documentación: La fase de documentación se ha dividido en dos subfases. La primera ocurrió de forma paralela en el tiempo al resto de fases. Durante la realización de estas se recogió gran cantidad de información, ya fuera respecto al estudio de las tecnologías, las tablas de requisitos y casos de uso o los diagramas de diseño. La segunda sub fase empezó tras la finalización de la fase de pruebas una vez que la aplicación se dio como finalizada, y consistió en reunir y organizar toda esa información para la creación de esta memoria. Esta fase tuvo una duración de 192 horas repartidas en los 163 días que duro todo el proyecto.

1.5 Medios empleados

Para el desarrollo del proyecto se han utilizado una serie de medios tanto hardware como software que explicare a continuación.

1.5.1 Hardware

- Hardware para desarrollo del proyecto.
  - Ordenador personal para la realización de todas las fases del proyecto y ejecución de la aplicación.
- Hardware para copias de seguridad
  - Disco duro externo para realizar copias de seguridad.
- Hardware para documentación
  - Impresora laser para la obtención de la documentación en papel.

1.5.2 Software

- Entorno de desarrollo
  - NetBeans Versión 6.9.1: Para la programación del código fuente de la aplicación.
  - Portal Pack: Es un plugin para NetBeans que añade una nueva interfaz para crear, configurar y trabajar con portlets.
  - JDK 6: Necesario para desarrollar y ejecutar el código Java.

- Servidores
  - GlassFish Server: Se utiliza como servidor de aplicaciones. Será el encargado de correr la aplicación creada. Sobre este servidor se ejecutara Liferay.
  - Liferay Portal: Se incluye dentro del apartado de servidores ya que así lo considera NetBeans al desplegarlo, sin embargo no es más que una aplicación WAR que corre sobre el servidor de aplicaciones y que contiene y gestiona los portlets.
  - MySQL Server: Sistema gestor de base de datos que gestiona las bases de datos utilizadas.

- Librerías
  - Twitter4J: Es una librería Java no oficial para la API REST de Twitter. Proporciona funciones de más alto nivel para las diferentes opciones que

ofrece Twitter a los desarrolladores en cuanto a manejar y gestionar usuarios, mensajes, contenido multimedia, etc.
    - Google Feeds API: Es una librería JavaScript de Google que permite descargar cualquier Atom público o fuente de medios RSS usando únicamente JavaScript.
    - JQuery: Es una librería que facilita el desarrollo de código JavaScript, ofrece diferentes funcionalidades para facilitar la interacción con el código HTML, inclusión de efectos, manejo de eventos y funciones para desarrollar con AJAX.
    - JQuery UI: es una extensión para JQuery que ofrece gran cantidad de widgets y efectos para la creación de interfaces de usuario.
    - Google Maps API: La librería de Google para añadir y desarrollar con los mapas que ofrece.
    - JDBC: Se utiliza para la conexión a la base de datos.

- Software de documentación
    - Microsoft Office 2007: Para la realización de la memoria.
    - StarUML: Para la realización de los diagramas UML.
    - Microsoft Power Point 2007: Para la realización de la presentación y algunas ilustraciones de la memoria.

1.6 Estructura de la memoria

En este apartado se incluyen los documentos completos de casos de uso y requisitos de software obtenidos en la fase de análisis. Para facilitar la lectura de la memoria, se incluye a continuación un breve resumen de cada capítulo.

Capitulo 1 – Introducción y objetivos.

Recoge una visión general sobre el proyecto, incluyendo un resumen sobre el problema a resolver, los objetivos planteados tanto de la aplicación como personales, las fases llevadas a cabo y los medios empleados para la realización de la aplicación.

Capitulo 2 – Estado del arte.

En este capítulo se expone una visión general de cómo está el mercado en el ámbito sobre en el que se centra el proyecto. Se analizan aplicaciones similares y posibles tecnologías a utilizar. Finalmente se explican las principales tecnologías elegidas para la resolución del problema planteado.

Capitulo 3 – Entorno de desarrollo.

Se describen las características de Hardware y Software que se han utilizado para el estudio, desarrollo y ejecución del proyecto, incluyendo la información sobre las versiones utilizadas. Además se incluye una guía sobre los diferentes pasos a seguir para la instalación y configuración del entorno de desarrollo.

Capitulo 4 – Análisis del sistema.

En este capítulo se realiza un resumen de los casos de uso y requisitos de software recogidos en la fase de análisis. Los documentos completos se podrán ver en los apéndices al final de la memoria.

Capítulo 5 – Diseño del sistema.

En este capítulo se recogen y explican los diagramas obtenidos en la fase de diseño. Se expondrá la arquitectura del sistema y el diseño estático y dinámico de los componentes desarrollados. Finalmente se expondrá el diseño de la base de datos utilizada.

Capitulo 6 – Pruebas

En este capítulo se recoge el plan de pruebas realizado para comprobar que la aplicación funciona como debe.

Capitulo 7 – Conclusiones y trabajo futuro.

En este capítulo se recogen las conclusiones obtenidas a la finalización del proyecto, se analizan los objetivos planteados al inicio para comprobar si se han cumplido, se ofrece información adicional como por ejemplo las dificultades encontradas y se exponen posibles trabajos futuros que se podrían realizar sobre el proyecto.

Capitulo 8 – Presupuesto.

El último capítulo de la memoria es el presupuesto del proyecto. En este capítulo se hace un resumen de la planificación y duración de cada fase, donde se incluye el diagrama de Gantt. El coste total se obtiene de calcular los costes de personal, elementos hardware y software y de los materiales fungibles.

Glosario

Se exponen todos los acrónimos que se pueden leer en la memoria junto con su significado.

Referencias

Se listan las fuentes bibliográficas a las que se hace referencia en la memoria.

Apéndices

En este apartado se incluyen los documentos completos de casos de uso y requisitos de software obtenidos en la fase de análisis.

Capítulo 2

Estado del Arte

2.1 Introducción

Como he explicado en el capítulo primero, el proyecto consiste en un portal de sindicación de información deportiva que se mantendrá actualizado de forma automática, simplemente accediendo a los recursos puestos a disposición por los diferentes medios.

En este capítulo se hará un resumen general del problema a resolver, ofreciendo una visión del estado actual de las tecnologías y de las aplicaciones similares que se pueden encontrar en el mercado.

Una vez hecho esto, se expondrá la solución al problema planteado en el proyecto, indicando las tecnologías que se van a utilizar y explicando estas de una forma más detallada. De cada tecnología se explicará qué ofrece, cómo funciona, sus ventajas y desventajas.

2.2 Estado del arte

Desde la época en la que los periódicos sólo se editaban en papel, hasta la actualidad donde el auge de Internet ha permitido la digitalización de dichos diarios, no sólo ha cambiado la forma en la que los usuarios acceden a la información; desde la comodidad de su hogar sin necesidad de bajar al quiosco, de una forma instantánea y actualizada al segundo, sino que se han ido añadiendo nuevas soluciones informativas enriquecedoras y de interacción con los lectores, apoyándose en las múltiples posibilidades de las nuevas tecnologías y de Internet.

En un principio los periódicos editados en papel eran junto con la radio y la televisión uno de las principales fuentes de información para las personas. Con el auge de Internet y la digitalización de los diarios, la versión en papel ha perdido fuerza debido en gran medida a que las versiones digitales son gratuitas, pero también a que los periódicos en papel únicamente informan, no permiten ninguna interacción por parte del lector.

Un periódico sigue una jerarquía respecto al tipo de noticia y la importancia de ésta. Para sacar la tirada diaria del periódico se necesita una serie de personal que también sigue una estructura jerárquica desde el director del periódico hasta el personal de documentación pasando por los subdirectores, redactores jefes, y los redactores o periodistas.

Las versiones digitales también siguen esta jerarquización pero, debido a las posibilidades que ofrecen las nuevas tecnologías es posible presentar la información de una manera mucho más personalizada, al gusto del lector y añadiendo nuevas soluciones que le dan a estas versiones un valor añadido con respecto a sus versiones en papel.

La digitalización de los periódicos no comenzó hasta mediados de los noventa [1] ya que no se veía claro el futuro de la prensa en internet. En un principio se pensaba en la digitalización como una simple forma de aumentar la audiencia, que la información llegara a muchas más personas de una forma mucho más fácil, Con el tiempo se ha demostrado que la digitalización permite que los lectores accedieran a la información de una forma mucho más interactiva, ya no sólo limitándose a leer una simple copia de las noticias en una pantalla. Ya no se ofrecen sólo noticias escritas sino que se puede acceder a otro tipo de recursos basándose en las herramientas que ofrece Internet y las nuevas tecnologías, tales como videos, audios, gráficos, etc.

Además de permitir el acceso a la información a través de la página web de los periódicos, también se han creado otras formas para que los usuarios puedan acceder a dicha información, como son las fuentes RSS, que permiten acceder a la información sin tener que acceder necesariamente a los diferentes sitios web de los periódicos.

En los últimos años, con la gran explosión de las redes sociales, los usuarios ya no tienen únicamente el rol de consumidor de información sino que ahora poseen también el rol de redactor o reportero, pudiendo crear sus propias noticias de lo que ocurre en su

ciudad o incluso en su barrio y compartirlas con el mundo entero. Es por esto por lo que se han hecho tan populares como indispensables para muchísimas personas. Las redes sociales se crearon en un principio para poder comunicarse de una forma diferente e interactiva con nuestros conocidos, pero han pasado de ser simplemente una herramienta de comunicación social a ser una fuente importantísima de información en sí misma, incluso más rápida que los propios periódicos digitales. Mientras que un periódico tiene un número limitado de reporteros y redactores, las redes sociales interconectan a gente de todo el mundo pudiendo saber lo que ocurre en cualquier lugar casi instantáneamente.

Distanciándose de la prensa digital pero continuando en el tema de la interacción y el intercambio de información existe otra tecnología que permite comunicar aplicaciones entre sí y aplicaciones con usuarios para cooperar o para intercambiar información. Estos son los Servicios Web y están muy integrados en la Web hoy en día, por ejemplo, las redes sociales permiten el acceso a su información a través de estos servicios.

2.3 Planteamiento del problema

Normalmente, cuando queremos obtener información sobre un tema que nos interesa accedemos a internet y buscamos páginas web, blogs y foros que traten sobre él. Normalmente consultamos siempre las mismas páginas, y las que más nos interesan las guardamos en los marcadores de nuestro navegador. Cuando queremos volver a verlas vamos visitando una a una cada una de ellas, pero existen otras formas de facilitar su lectura.

El principal objetivo del proyecto es la creación de un sitio web de sindicación de información deportiva que obtenga ésta de diferentes fuentes y las disponga de tal forma que los usuarios puedan acceder a ella de una forma centralizada y rápida sin tener que visitar las diferentes fuentes individualmente.

Un sindicador de información es un tipo de software que recoge información de distintas fuentes web actualizando el contenido automáticamente cuando estas fuentes cambian. Se podría pensar que no hay apenas diferencia entre la página web de un periódico cualquiera y el sitio web de sindicación de información propuesto como proyecto, pero no es así. La principal diferencia entre ambos es que mientras el periódico redacta su propia información y después la dispone en su página web, en el sitio web sindicador de información no habrá información propia, como tampoco habrá redactores, simplemente se obtendrá la información ya redactada que las diferentes fuentes ponen a disposición de los usuarios. Cuando un usuario esté interesado en una de las noticias y desee leerla al completo será redirigido a la fuente original de la misma. Su principal ventaja es la automatización, una vez creado se actualizará automáticamente sin necesidad de que nadie escriba nuevas noticias.

Lo que se pretende es que el sitio web a desarrollar se distinga de un diario online [2] y [3]. Se busca que el usuario tenga mayor posibilidad de personalización y que pueda elegir por si mismo qué información le interesa y cuál no, cosa que no puede hacerse en

los diarios digitales. Teniendo esto en cuenta, se han buscado en Internet ideas y posibles soluciones a cómo resolver el problema planteado. Entre las posibilidades analizadas se han encontrado aplicaciones interesantes, similares a lo que se pretende realizar y que han ayudado a centrar el objetivo en algo más concreto.

El primero de los sitios web a los que se quiere hacer referencia es el antiguo y ya desaparecido portal de Google *IGoogle*. En él se ofrecía una serie de información de diferentes tipos dividida en *widgets*. Un widget es una mini aplicación que tiene una utilidad concreta. Cada uno de estos widgets recogía información de una fuente independiente del resto, lo que permitía al usuario añadir únicamente los widget con información que le resultará interesante y además mantener la estructura de la página que deseará. Buscando aplicaciones similares en la actualidad se han encontrado otros ejemplos parecidos.

La web de Garmin Connect que recoge información de entrenamientos personales obtenidos de aparatos como pulsómetros y GPS tiene una estructura similar. La información que se muestra está dividida en ventanas independientes las unas de las otras. [4].

La web de la ACB [5], aunque no permite modificar la disposición de la página, maqueta la información en pequeñas ventanas independientes, siguiendo una idea similar a lo ya comentado.

La interfaz Metro de Windows 8, que sigue una estructura similar ofreciendo diferente información en apartados independientes.

Por último, el portal Netvibes [6] que es muy similar a lo que se pretende desarrollar. Permite añadir una serie de ventanas o widgets con información variada, permitiendo al usuario añadir tantas ventanas como quiera y elegir su distribución.

Tras analizar todos los ejemplos comentados, se ha decidido realizar un sitio web de sindicación de información en forma de componentes tipo widget de tal forma que los usuarios tengan opciones de personalización. Además se van a utilizar algunas de las diversas tecnologías existentes que permiten obtener información de la web para crear los componentes que se ofrecerán a los usuarios.

2.4 Propuesta de solución

En este apartado se concretarán las tecnologías seleccionadas como solución para el desarrollo del proyecto así como el análisis previo que se realizó para la elección de las mismas.

Cuando comencé a pensar en posibles soluciones para el objetivo concretado en el apartado anterior, lo primero que me vino a la mente fue qué lenguaje de programación utilizar para el desarrollo del proyecto. Me decanté por Java. Elegí Java porque era un lenguaje que no conocía. En un principio lo vi como una desventaja porque tendría que

dedicar un tiempo a aprenderlo antes de empezar con el proyecto, pero también pensé que sería un tiempo bien empleado, ya que Java es uno de los lenguajes de programación más utilizados en la actualidad y a la larga tendría que utilizarlo.

Para aprender Java utilicé dos libros, el primero llamado Aprendiendo Java en 21 días [7]. Este es un buen libro para aprender los primeros pasos de este lenguaje orientado a objetos. Quizá a día de hoy haya quedado un poco desfasado en alguno de sus apartados pero es una muy buena forma de empezar si no se conoce nada de Java. El segundo libro que estudié, más actualizado que el anterior fue el llamado Java 7 [8]. Este libro también está muy bien preparado para comenzar a adquirir conocimientos de Java. Me pareció que el autor explicaba de una forma muy clara y sencilla de aprender. Elegí este libro principalmente porque contenía un apartado donde explicaba cómo trabajar con servicios web, una de las tecnologías principales que quería utilizar en el desarrollo del portal de sindicación de información.

Una vez concretado el objetivo a desarrollar y conocido el lenguaje de programación a utilizar me centré en buscar diferentes posibilidades que me permitieran crear un sitio web basado en componentes tipo widget o similares. Concretamente me centré en estudiar los llamados portales web.

Un portal web [9] se define como una plataforma de software para la creación de aplicaciones y sitios web adaptables por los usuarios, que ofrezcan acceso rápido y centralizado a una serie de recursos y servicios provenientes de diferentes fuentes, relacionados con un mismo tema. Los portales pueden incluir todo tipo de recursos: enlaces, buscadores, foros, documentos, aplicaciones de toda índole tales como sistemas de compra electrónica, gestores de correo electrónico, juegos, etc.

De esta forma, mediante un portal web se puede crear cualquier tipo de sitio web, ya sea un sitio web de colaboración, que incluya herramientas como blogs, wikis, compartición de documentos; o aplicaciones sociales, de comercio electrónico o como el que se va a desarrollar, un sitio web de integración.

Entre las diferentes soluciones que ofrecían la posibilidad de crear un sitio web basado en componentes modulares, me centré en tres, Jboss Portal [10], Liferay Portal [11] y Netvibes . Los dos primeros son portales web que soportan la especificación JSR 286, para el despliegue y creación de componentes Portlets. Los portlets son aplicaciones web modulares desarrolladas en Java que ocupan una porción de una página web en lugar de la página completa.

Netvibes en cambio no es un portal web como tal. Es una herramienta web que, al igual que los portales web comentados anteriormente, permite la creación de un espacio personalizado en el que agregar componentes, pero en este caso componentes widgets. Su funcionalidad es prácticamente la misma que la de los portales web ya que permite acceder a la información deseada por el usuario de forma centralizada en lugar de estar visitando cada sitio web por separado. La principal diferencia entre ellos es que los portales web se basan en Portlets que están modelados para ser ejecutados en el lado del servidor, mientras que Netvibes se basa en Widgets modelados para ejecutarse en el lado del cliente [12]. Por esta razón descarté el uso de Netvibes, ya que como comenté antes, el proyecto va a ser desarrollado en Java. Los widgets deben desarrollarse en un lenguaje para el lado del cliente tipo JavaScript, es decir que se ejecute sobre el navegador. En el

apartado siguiente tecnologías explicaré con más detalle qué son los portlets, lo componentes finalmente seleccionados para el desarrollo del proyecto.

El siguiente paso fue buscar la forma de obtener información de internet para poder crear mi portal. Tenía que encontrar tecnologías que me permitieran obtener información, procesarla y mostrarla a través de mis portlets. Mi primera idea fue pensar en Google News. El servicio de noticias de Google recopila artículos de los diferentes sitios web de noticias del mundo y los muestra centralizados en el propio navegador del usuario. Pensando en ello se me ocurrió que podría realizar algo parecido utilizando la tecnología RSS que ofrecen casi todo los diarios digitales y blogs en la actualidad.

Un RSS, cuyas siglas significan Really Simple Syndication [13], es un archivo XML con un formato específico que es generado por un sitio web con una versión reducida de la información publicada en él. Estos archivos son utilizados para compartir la información del sitio web con los usuarios sin que estos tengan que visitarlo salvo que se desee leer la información completa. Estos archivos se modifican automáticamente cuando se realiza alguna actualización en el sitio web.

Para leer estos archivos es necesario el uso de un lector RSS. Un lector RSS es un programa que reúne todos los artículos de las fuentes RSS a las que un usuario este suscrito y los muestra de forma legible para éste. Es similar a un gestor de correo electrónico salvo que en lugar de consultar periódicamente el buzón para mostrar los mensajes, accede periódicamente a los archivos RSS a los que se esté suscrito para obtener la última versión de estos. En internet existen muchos lectores RSS, tanto online a través del navegador, como de escritorio, listos para instalar y utilizar.

En mi caso, teniendo en cuenta el ejemplo de Google News, decidí crear mi propio componente portlet lector de RSS que permita visualizar centralizadamente fuentes RSS de diferentes sitios web, permitiendo al usuario seleccionar la fuente RSS que desee leer de entre las opciones que serán ofrecidas. Para la visualización de los artículos de cada una de las fuentes RSS voy a intentar programar un slideshow parecido al de la web *defensacentral.com* [14].

Buscando otras posibles fuentes de donde obtener información se me ocurrió que las redes sociales podrían ser un buen sitio a analizar. Actualmente las redes sociales son uno de los mayores flujos de información existentes. Con las redes sociales ya no sólo se puede estar informado por medio de los diarios digitales, sino que cualquier usuario puede ser un reportero que ofrezca información actualizada al segundo en cualquier ámbito imaginable, incluso hasta las noticias de su propio vecindario. Por estas razones me decanté por estudiar la forma de acceder a la información que se mueve en las redes sociales. Analizando el apartado para desarrolladores de varias de las redes sociales actuales descubrí el modo en el que éstas ofrecen la información a cualquier persona que quiera programar su propia aplicación. Utilizan los llamados Servicios Web [15]. Definidos de forma muy general los servicios web son una tecnología que permite interoperar e intercambiar información entre aplicaciones utilizando protocolos y estándares abiertos.

Como las redes sociales son una fuente perfecta para obtener información para utilizar en mi portal y la mayoría de ellas se basan en los servicios web para ofrecerla a los desarrolladores, me dediqué a estudiar en profundidad que es un servicio web y como

trabajar con ellos. Cuando lo hice me di cuenta de que las redes sociales se basan en uno de los dos estilos de diseño que existen a la hora de manejar servicios web, los servicios web basados en REST [16], [17], pero que existe otro estilo basado en SOAP [17] y [18]. Después de realizar el estudio sobre los servicios web decidí que incluiría ambos estilos en el desarrollo de mi portal. Los servicios web, junto con sus dos estilos de diseño serán explicados detalladamente en el apartado Tecnologías más adelante.

Con los servicios web basados en REST voy a realizar un portlet lector de tweets que muestre información en tiempo real de los usuarios de Twitter que me interesen para el contenido del portal. En cuanto a los basados en SOAP, voy a programar varios portlets que ofrecerán información sobre el Mundial de Fútbol de Brasil 2014.

Debido a que el portal va a ser un sindicador de información deportiva y que quiero añadir estadísticas de la liga de fútbol profesional, he buscado la mejor manera de integrarlas en mi portal. La LFP no da ninguna opción de obtener las oficiales, de modo que he decidido crear mi propio servicio web en el que iré modificando las estadísticas después de cada jornada como hace la propia LFP salvo por que yo las ofreceré mediante el servicio. De esta forma no sólo me centraré en obtener información de los servicios web sino que también aprenderé a crearlos. Después en el portal crearé varios portlets que lo consuman.

Por último he decidido analizar el posible uso de la tecnología AJAX [19] en alguno de los portlets para ofrecer mayor dinamismo evitando tener que realizar cargas completas de página al interactuar con ellos. Esta tecnología también será explicada en el apartado Tecnologías.

2.5 Tecnologías

2.5.1 Portlets

Un portlet [9] es un componente modular basado en tecnología Java que procesa pedidos y genera contenido específico. Los portlets son manejados y visualizados en las páginas de un portal web. Una vez desplegados los portlets se observan como pequeñas ventanas que contienen diferente información, pueden contener tanto aplicaciones web como simples páginas web en formato HTML. Estas ventanas se sitúan unas encima de las otras y también a los lados pero sin solaparse.

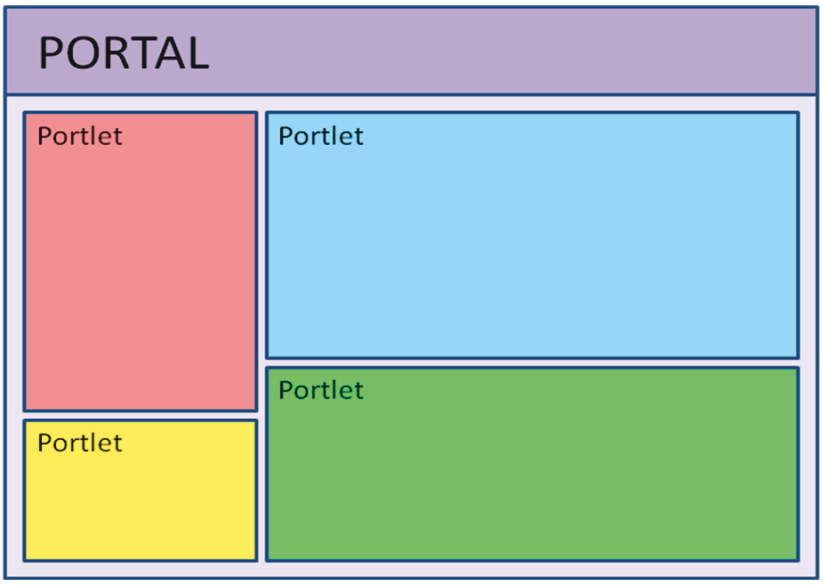

Ilustración 1. Portlets en portal.

En un principio, cada proveedor de portales definía la forma de desarrollar componentes para su plataforma, lo que provocaba que cada componente tuviera dependencia directa con el portal para el que habían sido desarrollados. Esto era un problema ya que se pretendía que todos los componentes fueran portables entre plataformas. Para dar solución al problema, en 2003 los principales proveedores de portales se pusieron de acuerdo en crear un estándar para desarrollar sus componentes. Crearon el estándar JSR-168 [20] y todo portlet que hubiera sido desarrollado bajo esta especificación sería desplegable en cualquier portal que también siguiera el estándar. En 2008 se definió el estándar JSR-286 [21] que es el estándar actual de los portlets.

Entre los aspectos incluidos es estos estándares se define tanto el modo de ejecución de un portlet y el ciclo de vida de los mismos, como los estados y modos, el empaquetado y el despliegue de estos.

Para que los portlets se puedan ejecutar es necesario un contenedor de portlets que los provea del ambiente de ejecución requerido y que se encargue de controlar el ciclo de vida de los mismos. También se encarga de mantener de forma persistente las preferencias de cada portlet y de recibir las peticiones que desde el portal se envían para que se ejecuten en el portlet alojado correspondiente (mediante el paradigma request/response). El contenedor no es el encargado de agregar el contenido generado por los diferentes portlets a la página del portal. Simplemente devuelve el trozo de código (llamado fragmento) al servidor del portal y es este el encargado de mostrarlo en el espacio asignado para el portlet.

Como señalé en el párrafo anterior el contenedor también se encarga de controlar el ciclo de vida de los portlets. El ciclo de vida de los portlets se basa en el paradigma de petición y respuesta, es decir, recibe peticiones de diferentes tipos y devuelven código generado hacia el portal. Después de ejecutar la petición, los portlets generan código únicamente para una parte de la página web, por lo que el ciclo de vida de cada portlet debe coordinarse con el del resto.

El ciclo de vida de un portlet tiene varias fases: inicialización, visualización, procesamiento de acciones, procesamiento de eventos y destrucción.

La fase de inicialización es ejecutada por el contenedor de portlets mediante el método Init(). Esta fase sólo es ejecutada una vez en todo el ciclo de vida de un portlet. Sirve para inicializar el portlet que va a ser desplegado.

La fase de visualización es ejecutada por el contenedor de portlets mediante el método Render() o alguno de los métodos más específicos delegados por este doView(), doEdit() y doHelp(). En esta fase se genera código HTML que se devolverá al portal para que este la visualice.

La fase de procesado de acciones es ejecutada mediante el método processAction(). Esta fase es ejecutada cuando se realiza una petición de cambio sobre uno de los portlets desplegados en el portal y que reflejará un cambio sobre el portlet en la fase de visualización. Después de esta fase siempre se ejecuta la fase de visualización para todos los portlets que estén desplegados en la página.

La fase de procesamiento de eventos es en la que el contenedor de portlets gestiona los eventos para los que está escuchando. La principal función es el manejo de eventos para la interconexión entre portlets. Después de esta fase también se ejecuta siempre la fase de visualización.

Por último la fase de destrucción que es ejecutada mediante el método Destroy(). Es ejecutada por el contenedor de portlets cuando el portlet va a ser quitado de la página para indicar al portlet que queda fuera de servicio.

En el siguiente diagrama se puede observar lo explicado sobre el ciclo de vida de los portlets de forma gráfica mediante un ejemplo.

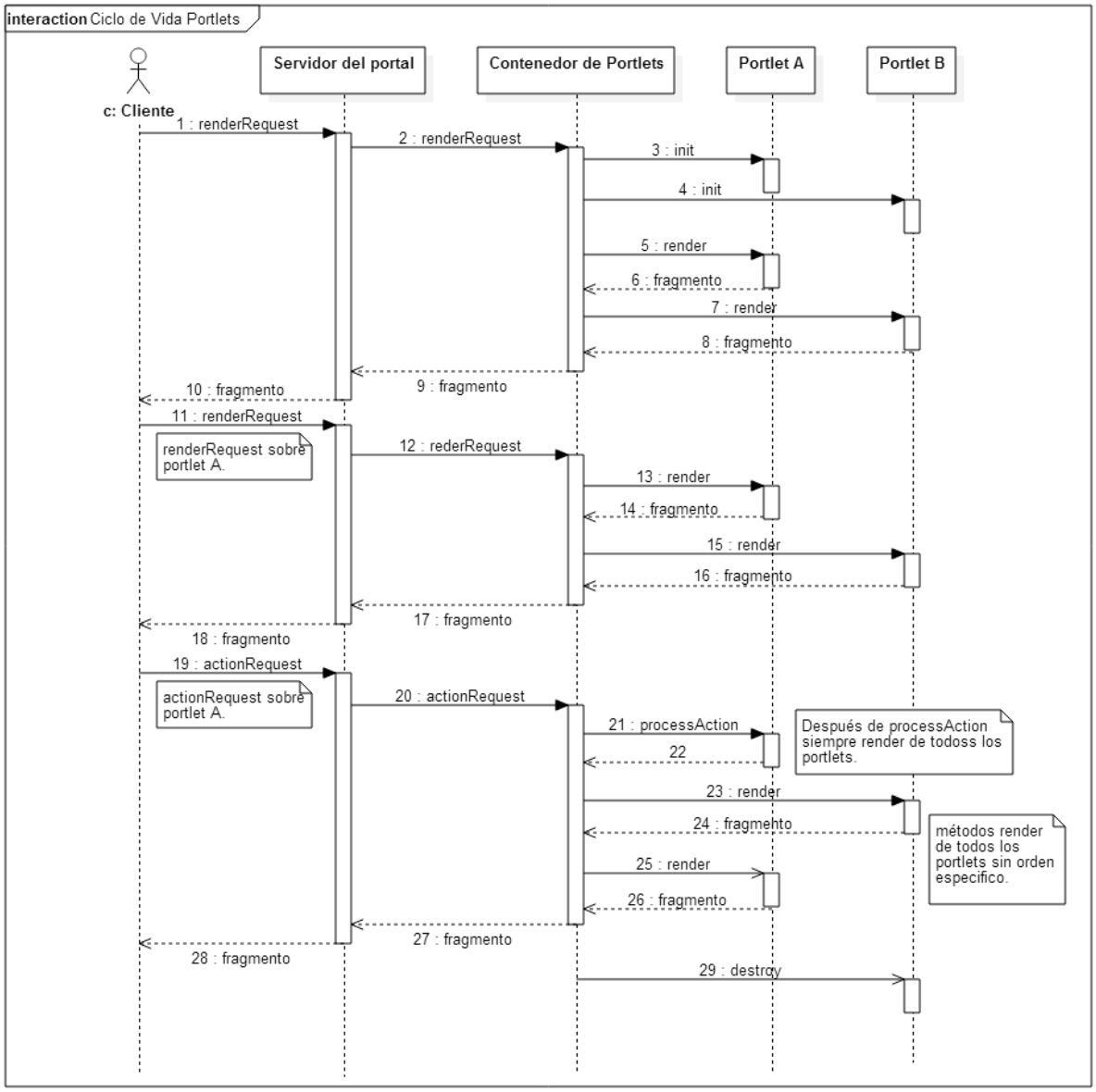

Ilustración 2. Ciclo de vida de los portlets basado en imagen del libro Portlets in action [9].

Cuando se añade un portlet al portal o se accede a la página del portal con el portlet ya cargado, el contenedor de portlets inicia la fase de inicialización con el método init() en cada uno de los portlets.

Seguidamente se inicia la fase de visualización de los portlets, tanto de los nuevos como de los ya añadidos en el portal. Mediante el método render() o el método más específico doView() delegado por éste, el contenedor de portlets pide a los portlets que generen el fragmento de código a mostrar y estos lo devuelven al contenedor. El contenedor se encarga de devolver el fragmento de código al servidor del portal para que este los muestre en el Portal.

En caso de que el servidor del Portal reciba una petición de visualización posterior de la inicial, éste le pasará la petición al contenedor para que se encargue de nuevo. Una petición de visualización de este tipo se da cuando se quiere cambiar de vista en el portlet

pero sin tener realizar ninguna acción previa, por ejemplo al pinchar un enlace a otra sección de información o cuando se desea cambiar entre los modos edición, vista y ayuda de un portlet. Los modos de un portlet son explicados en esta misma sección más adelante. Hay que tener en cuenta que cuando se hace una petición sobre un portlet concreto, siempre se realiza la fase de visualización de todos los que estén desplegados en la página del portal, independientemente de sobre cuál haya sido hecha la petición.

Cuando el servidor del Portal recibe una petición de acción, la pasa al contenedor para que este lance el método processAction() del portlet sobre el que se ha hecho la petición. Un ejemplo de acción puede ser el procesar la información de un formulario o introducir datos en una base de datos. Cuando el método processAction() ha finalizado, el contenedor debe lanzar el método correspondiente de la fase de visualización de todos los portlets desplegados en el portal, de nuevo independientemente de sobre cuál haya sido hecha la petición. Cabe decir que los métodos de visualización de todos los portlets se ejecutan sin un orden específico, pueden ejecutarse incluso en paralelo.

Finalmente, en el caso de que se quiera eliminar un portlet de la página del portal, el contenedor de portlets lanza la fase de destrucción de ese portlet ejecutando el método destroy().

Como comente durante la explicación del diagrama del ciclo de vida, los portlets poseen diferentes modos de funcionamiento, Los modos definen las diferentes formas con las que los usuarios pueden actuar con el portlet. Todos los portlets tienen tres modos predefinidos, el modo *view* es el modo por defecto. Cuando el portlet está en modo view se visualiza el propósito general del portlet, por ejemplo en un portlet que gestione el correo electrónico, el modo view permitiría leer y enviar mensajes. Los otros dos modos predefinidos por la especificación son los modos *edit* y *help*. Estos modos pueden ser accedidos por el usuario en cualquier momento del ciclo de vida del portlet. El modo edit permite a los usuarios hacer configuraciones en el comportamiento de la aplicación. En el ejemplo del correo electrónico una acción a realizar podría ser el recordatorio o cambio de contraseña o la configuración del tiempo de actualización de la bandeja de entrada. Cualquier anotación para ayudar al usuario a manejar el portlet iría incluida en el modo help del portlet.

Otra característica que poseen los portlets es la de el *estado de la ventana*. El estado de la ventana especifica el espacio que el portlet va a ocupar en pantalla. Existen tres estados predefinidos, *normal*, *maximizado* y *minimizado*. Si se pulsa el estado maximizado el portlet ocupará todo el espacio disponible. En el caso de pulsar el estado minimizado sólo se mostrará la barra de titulo del portlet.

En cuanto al empaquetado y despliegue de los portlets se realizan mediante archivos WAR. Un archivo WAR contiene toda la información necesaria para el despliegue y en correcto funcionamiento de una aplicación web. Este paquete contendrá las clases java del portlet, los JSP, archivos JavaScript y CSS utilizados, las imágenes, las librerías y los documentos XML de despliegue. Colocando el archivo WAR de un portlet en la carpeta de despliegue del portal elegido, este quedará desplegado automáticamente y disponible para mostrarse en la páginas del portal.

2.5.2 Servicios Web

2.5.2.1 Introducción

En la actualidad existen una gran cantidad de tecnologías para el desarrollo de aplicaciones Web, muchas de ellas incompatibles entre sí, por lo que pueden existir problemas a la hora de integrar aplicaciones de diferentes organizaciones que deban trabajar conjuntamente intercambiando información, entre los sistemas de información de una misma organización o incluso entre las propias organizaciones y los sistemas de software utilizados por sus clientes.

Para poder llevar a cabo esta cooperación o intercambio de información, los desarrolladores de todas las partes deberían ponerse de acuerdo en cuanto a las tecnologías a utilizar. Esto haría muy compleja la cooperación y en ocasiones imposible si fueran más de dos las entidades que tuvieran que cooperar y necesitaran utilizar forzosamente diferentes arquitecturas de componentes y tecnologías para crear sus aplicaciones. Para solucionar estos problemas surgió el concepto de Servicio Web.

El W3C, en la nota del grupo de trabajo del 11 de Febrero de 2004 [15], define los servicios Web como *sistemas de software diseñados para permitir interoperabilidad máquina a máquina a través de una red. Estos servicios tienen interfaces descritas en un formato procesable por las máquinas que son accesibles a través de una red, como por ejemplo internet. Permiten a los clientes interoperar con los servicios, los cuales se ejecutan en el sistema que los aloja.*

Es decir, estos sistemas de software permiten la interoperabilidad y el intercambio de información entre aplicaciones, basándose en estándares abiertos (como XML) para la creación de los mensajes de comunicación y utilizando, protocolos de Internet (como HTTP, FTP o SMTP) para el transporte de los mismos. De esta forma se consigue que independientemente de la plataforma donde se ejecuten las aplicaciones (ya sea Windows, UNIX, Linux,…) y del lenguaje de programación concreto utilizado para crearlas (C++, Java, .NET,…), estas puedan comunicarse con otras aplicaciones a través de una red sin problemas.

Los Servicios Web están muy relacionados con el modelo de arquitectura orientada a servicios (SOA). SOA [22] [23] es un estilo de programación con el cual se pretende trasladar la idea de reutilización del código a la web para dar solución a los requisitos del negocio de forma distribuida. Se basa en el desarrollo del software mediante la programación de módulos autocontenidos y con funcionalidades muy reutilizables. Estos módulos interactúan con otros a través de la red para dar solución a los problemas a resolver. Por estas razones la mayoría de las veces suele identificarse a los Servicios Web como método de implementación en este tipo de arquitectura. No obstante, no es el único método, se pueden implementar usando cualquier otra tecnología basada en servicios (CORBA, DCOM, RPC).

Dentro de los Servicios Web existen dos estilos de diseño que se basan en este concepto pero que tienen diferencias significativas. Son los servicios web basados en SOAP y los Servicios Web basados en REST. A continuación se describirán cada uno de estos estilos.

2.5.2.2 Servicios Web basados en SOAP

¿En qué consisten?

Como se ha comentado en la introducción a los servicios web, estos sistemas de software permiten la comunicación e integración entre aplicaciones heterogéneas. Para ello se intercambian mensajes creados basándose en estándares abiertos como es XML utilizando protocolos de transporte de internet.

Este estilo de diseño de servicios web sigue estas especificaciones generales pero añadiendo características propias. La principal el uso de SOAP, que es un protocolo de comunicación basado en XML que describe el formato de los mensajes a intercambiar entre las partes involucradas cuando se invoca un servicio web. Estos paquetes SOAP se envían sobre un protocolo de transporte de internet como puede ser HTTP.

Para que los clientes del servicio web puedan entender que es lo que el servicio ofrece, como invocarlo y donde encontrarlo, debe existir un contrato que lo defina. Esto se realiza mediante el fichero WSDL también formado mediante XML.

Cuando un cliente desea acceder al fichero WSDL de descripción de un servicio, o el propio desarrollador del mismo desea publicarlo para permitir el acceso, ambos deberían utilizar el protocolo UDDI que implementa las funcionalidades necesarias para poder realizar ambas funciones.

En el apartado siguiente se definirá de forma más detallada los protocolos y estándares comentados.

Protocolos y estándares

SOAP

Es un protocolo definido por el W3C para el intercambio de información entre aplicaciones en un entorno distribuido. Establece el formato de empaquetado de los documentos XML (mensajes SOAP) que van a ser intercambiados entre los servicios Web y las aplicaciones clientes durante el proceso de comunicación, tanto para hacer peticiones como devolver respuestas.

SOAP aprovecha los estándares existentes, como XML para la codificación de los mensajes y HTTP como protocolo de transportes. De esta forma se hace posible la comunicación entre aplicaciones que se ejecutan en diferentes plataformas y desarrolladas en diferentes lenguajes.

Un mensaje SOAP no es más que un archivo XML que está compuesto por un elemento envoltorio (Envelope) y éste a su vez contiene un elemento cabecera (Header) y un elemento cuerpo (Body).

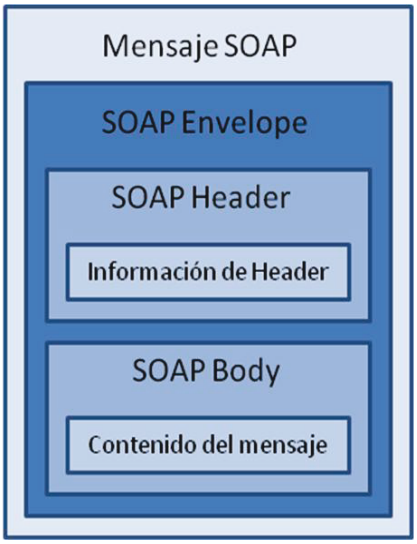

Ilustración 3. Mensaje SOAP.

El elemento Envelope es necesario y es el elemento raíz de un mensaje SOAP. Este elemento define el documento XML como un mensaje SOAP. Debe estas asociado al espacio de nombres "http://www.w3.org/2001/12/soap-envelope" que define el envoltorio como un SOAP envelope. Si no se usa este espacio de nombres el mensaje quedaría descartado.

El elemento Header contenido en el Envelope no es obligatorio pero puede incluir cualquier información necesaria para el procesado del mensaje en los nodos intermedios, como por ejemplo si estos nodos deben hacer algo con el mensaje, si es necesaria alguna autentificación, todo lo relacionado con certificados de seguridad, coordinación, etc.

En el elemento Header existen tres atributos que indican como un receptor del mensaje debe procesar el mismo. El atributo *role* indica quién debe procesar la cabecera. Puede tener los valores: *none* (indica que la información no debe ser procesada), *next* (indica que un nodo intermedio puede procesar la información, es posible que tenga que hacer algo con ella pero no es el destinatario final) y *ultimatereceiver* (indica que el destinatario final es el que debe procesar la cabecera). El atributo *mustUnderstand* indica si es obligatorio para un nodo que reciba el mensaje procesarlo o no. Por último el atributo *encodingStyle* define los tipos de datos usados en el documento y es necesario en cualquier elemento SOAP.

El elemento Body es requerido y es donde está contenido el mensaje real que debe llegar al destinatario final tanto en una petición como en una respuesta. Realmente no se especifica nada acerca del formato del contenido pero si que el destinatario debe poder entenderlo.

A continuación se muestra un ejemplo de comunicación mediante mensajes SOAP

```
    POST /InStock HTTP/1.1
Host: www.compralibros.com
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.compratuslibros.com/stock">
  <m:GetStockPrecio>
    <m:StockNombre>Don Quijote de la Mancha</m:StockNombre>
  </m:GetStockPrecio>
</soap:Body>

</soap:Envelope>


    HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.compratuslibros.com/stock">
  <m:GetStockPrecioResponse>
    <m:Moneda>Euro</m:Moneda>
    <m:Precio>19</m:Precio>
  </m:GetStockPrecioResponse>
</soap:Body>

</soap:Envelope>
```

El primer fragmento es una petición sobre un servicio web y el segundo fragmento es la respuesta. Como se puede observar el mensaje SOAP va dentro del código del protocolo de transporte.

La especificación SOAP define la estructura de los mensajes, pero no la forma en que se estos se intercambian, de esto se encargan los llamados enlaces SOAP (SOAP Bindings) Los enlaces SOAP definen los posibles métodos de transporte disponibles para acceder a una operación de servicio. Para poder enviar los mensajes se debe mapear SOAP con el protocolo de transporte correspondiente para después enviar el mensaje con las cabeceras propias del protocolo de transporte.

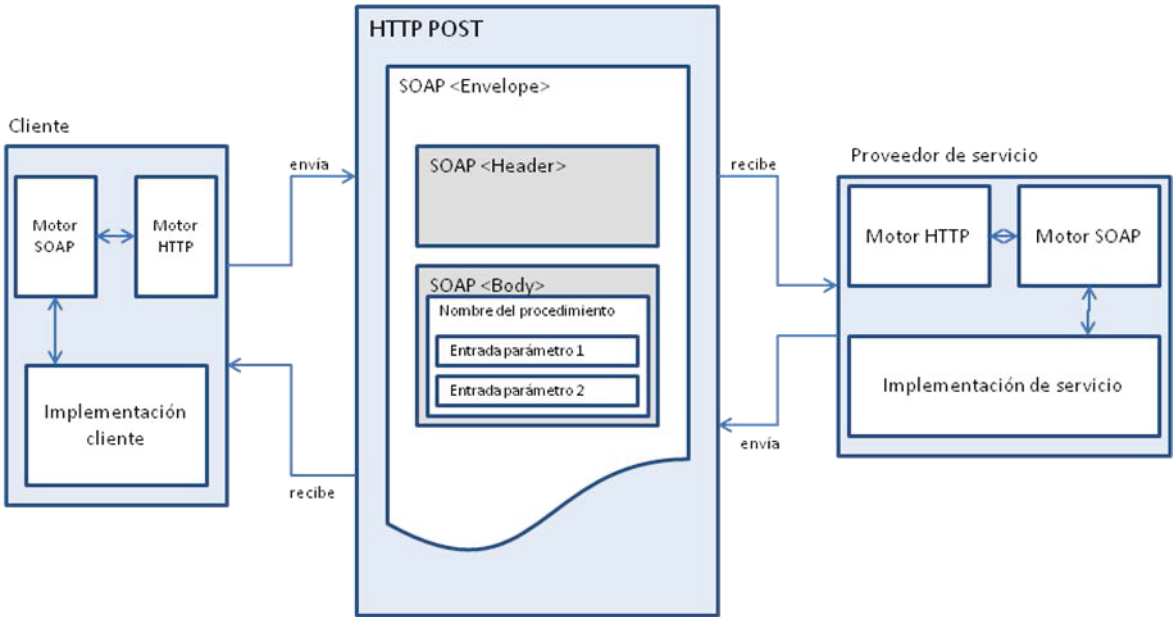

Ilustración 4. Mapeo SOAP - HTTP [23].

SOAP como tal no es adecuado por si sólo para implementar aplicaciones que incorporen características típicas de middlewares como pueden ser transacciones o envío fiable de mensajes pero tiene la ventaja de que se puede ir ampliando con extensiones tales como WS-Security y otras que no se han incluido en el estudio ya que no entran en el ámbito del objetivo final del proyecto.

WSDL

Es un documento escrito en XML que es usado para describir un servicio web. Este documento, que es creado cuando se publica el servicio, especifica la interfaz abstracta a través de la cual los clientes pueden acceder al servicio y los detalles de cómo se debe utilizar. Mediante este documento los clientes que desean acceder a la funcionalidad del servicio, simplemente necesitan interpretar el documento WSDL de modo que no es necesario conocer los detalles de implementación, la plataforma ni el lenguaje de programación utilizado.

Un documento WSDL está compuesto por un elemento raíz llamado definitions que a su vez está compuesto por los siguientes elementos:

Ilustración 5. Elementos de WSDL.

- *Types:* Esta sección define los tipos de datos usados en el servicio web y que están contenidos en los mensajes des siguiente punto.
- *Message:* Esta sección define los mensajes de entrada y salida en forma abstracta entre el servidor y el cliente. Cada mensaje puede contener una o más partes. Estas partes pueden ser de cualquiera de los tipos definidos en la sección Types. No hay que equivocar estos mensajes con los mensajes físicos que se envían en SOAP.
- portType: Contiene el conjunto de operaciones abstractas que soporta el servicio. Estas operaciones definen los tipos de mensajes a intercambiar entre el cliente y el servidor. Pueden ser de los siguientes tipos:
    - *Request-response*, en la que le cliente realiza una petición y el servidor envía la correspondiente respuesta.
    - *One-way*, en la que el cliente envía un mensaje pero no recibe una respuesta del servidor indicando el resultado del mensaje procesado.
    - *Solicit-response*, la contraria a la operación request-response. El servidor envía una petición y el cliente le envía de vuelta una respuesta.
    - *Notification*, la contraria a la operación one-way el servidor envía una comunicación del estilo documento al cliente.
- *Binding:* En esta sección se especifican los protocolos de comunicación y el formato de datos para una operación determinada.
- *Service:* En este elemento se informa sobre el punto de acceso a los servicios para cada uno de los protocolos definidos en el servicio a través de un elemento address.

La mayoría de aplicaciones de desarrollo cuentan con herramientas que facilitan la generación de los documentos WSDL a partir de código programado de un servicio y al contrario, desde el documento WSDL generan las interfaces necesarias para que un cliente pueda invocar el servicio en el lenguaje de programación deseado. Estas herramientas se basan en el uso de APIs preparadas para el desarrollo de servicios web como puede ser JAX-WS para el lenguaje de programación Java.

UDDI

Una vez desplegado el servicio web y especificada la forma de acceder a él mediante el documento WSDL, es necesario definir la forma en que el servicio se dará a conocer para que los clientes que lo deseen puedan descubrirlo y utilizarlo un sus aplicaciones. Para ello existe UDDI.

UDDI es un registro público independiente de la plataforma diseñado para publicar y descubrir información estructurada sobre empresas y los servicios que estas ofrecen. Estos datos pueden ser introducidos siguiendo diferentes tipos de clasificación para poder posteriormente encontrarlos en función de la categorización realizada (por empresa, por sector, etc.).

En el registro UDDI la información se clasifica en tres grupos de forma análoga a la que lo hacen las guías telefónicas: páginas blancas, páginas amarillas y páginas verdes.

- Las páginas blancas contienen información sobre las empresas como el nombre, la información de contacto, la descripción de la compañía, los servicios que ofrece, etc.
- Las páginas amarillas contienen información de las empresas organizadas por grupos de negocios, no por empresas concretas.
- Las páginas verdes contienen información técnica sobre los servicios que tiene la empresa. Es donde se encuentran los punteros a los WSDL de los servicios.

Cada entrada introducida en el registro UDDI contiene 4 atributos; *BussinessEntity* indica la empresa que realiza el registro de la entrada, *BussinessService* indica el servicio ofrecido, BussinessTemplate indica las diferentes formas de acceder al servicio expuesto y TModel que señala al WSDL del servicio.

UDDI define diferentes APIs para implementar las funcionalidades de publicación y búsqueda además de otras APIs para la gestión de información. Concretamente son seis:

- API de búsqueda para interrogar al registro en la búsqueda de servicios.
- API de publicación para añadir, modificar y eliminar entradas del registro.
- API de seguridad para controlar que no se realicen operaciones sin permiso como puede ser borrar un servicio de otro proveedor.
- API de custodia y transferencia de pertenencia para mover información o trasferir la custodia entre entradas del registro.
- API de suscripción para conocer los movimientos en el registro tales como nuevas entradas eliminaciones. En el caso de que un proveedor este suscrito recibirá toda esta información.
- API de réplica que da soporte para replicar registros y que estos estén sincronizados.

Todo lo comentado sobre UDDI es la teoría de cómo debería utilizarse. Los proveedores se registrarían y publicarían sus servicios. En un principio se hizo pensando que todos los servicios en el mundo estarían registrados pero la verdad es que actualmente muchos proveedores de servicios no registran sus servicios. En el caso de que los clientes deseen acceder al servicio de uno de estos proveedores deben contactar directamente con él y este le ofrece su documento WSDL.

¿Cómo funcionan?

Una vez descritos cada uno de los protocolos que forman parte de los servicios web basados en SOAP, es momento de mostrar el procedimiento que se lleva a cabo desde que el servicio se desarrolla hasta que el cliente lo consume. Esto se llevará a cabo mediante una de sus implementaciones más típicas, una arquitectura orientada a servicios. Se va a utilizar un ejemplo de esta arquitectura porque en ella se aprovechan al máximo cada uno de los protocolos descritos en el apartado anterior. Esto no quiere decir que

siempre que se usen servicios web se esté usando la arquitectura orientada a servicios y tampoco que siempre que se implemente una arquitectura SOA se tenga que usar servicios web SOAP.

En el siguiente esquema se puede observar el funcionamiento de los servicios web basados en SOAP.

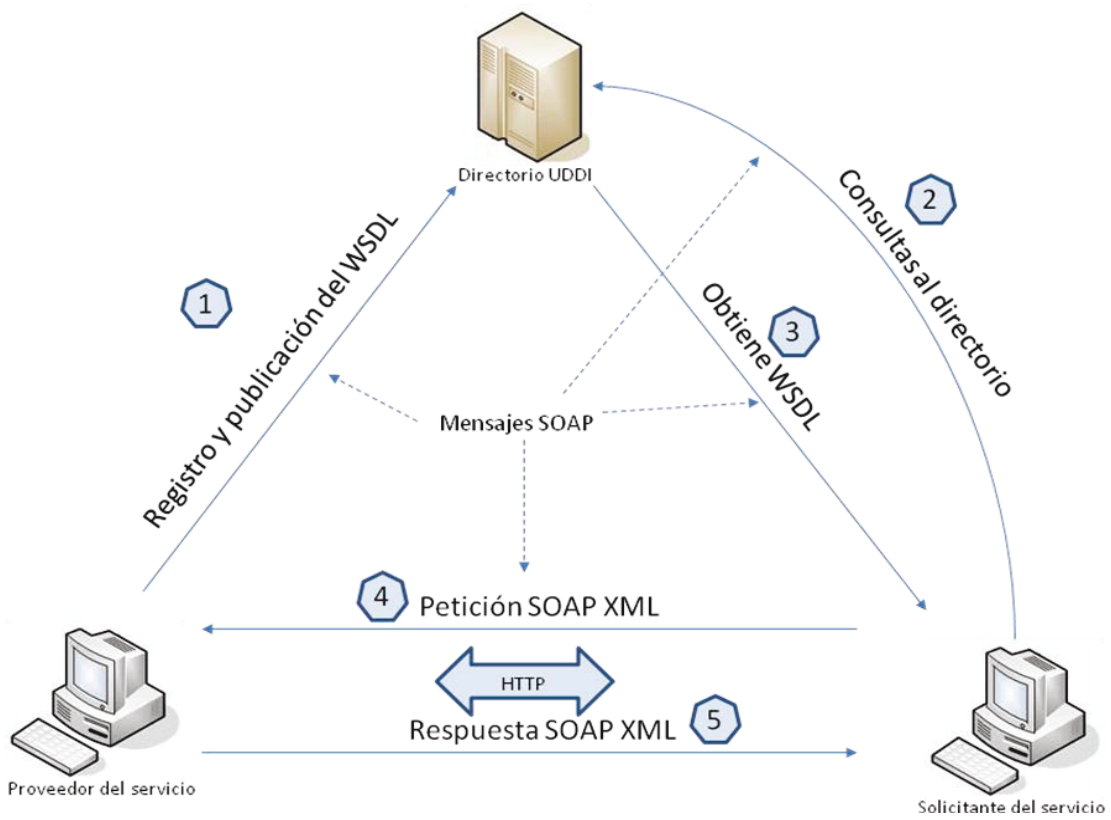

Ilustración 6. Funcionamiento servicio web SOAP.

Se puede observar que existen tres roles: el proveedor del servicio web, el cliente del servicio y el registro UDDI. La secuencia desde que el proveedor desarrolla el servicio hasta que el cliente lo consume es la siguiente:

1. Una vez el que se ha desarrollado el servicio web con la funcionalidad deseada y se ha generado el documento WSDL, el proveedor se inscribe en el registro UDDI y da de alta el servicio almacenando el documento descriptor de éste.

2. Un cliente hace una búsqueda en el registro UDDI acerca de servicios que se adecuen a sus necesidades.

3. El directorio UDDI devuelve el descriptor WSDL del servicio seleccionado. Éste será el mismo que el proveedor registró en el punto 1. Cabe indicar que la comunicación con el registro UDDI en todos los casos se realiza mediante mensajes SOAP.

4. El cliente, con el descriptor WSDL del servicio crea una aplicación cliente que consumirá el servicio web. La aplicación lo invoca mediante el envío de mensajes SOAP a través del protocolo de transporte elegido en los que indica que acción realizar y los datos de entrada necesarios.

5. El servicio web recibe la petición y ejecuta la funcionalidad. El servicio envía otro mensaje SOAP a la aplicación cliente con el resultado de la operación realizada.

Los mensajes intercambiados entre el servicio web y la aplicación cliente son mensajes SOAP enviados a través del protocolo de transporte, normalmente HTTP. Mucha gente opina que el uso de SOAP como protocolo de comunicación sobre HTTP que es un protocolo de transporte es innecesario ya que a través de POST de HTTP se podrían enviar los datos, sería como enviar un sobre SOAP dentro de otro sobre HTTP. De esta idea surge el segundo estilo de programación de los servicios web, los servicios web basados en REST que únicamente usan las operaciones del protocolo HTTP para el manejo de la información.

Toda la información explicada sobre servicios web basados en SOAP ha sido previamente estudiada de las siguientes fuentes [17] [18] [23].

Los servicios web basado en REST son el segundo estilo de programación estudiado y son descritos a continuación.

2.5.2.3 Servicios Web basados en REST

¿En qué consisten?

Como se ha comentado al final del apartado anterior de los servicios web basados en SOAP, este tipo de servicios web necesitan de su propio protocolo de comunicación para poder comunicar diferentes aplicaciones y éste a su vez se transfiere sobre otro protocolo de transporte como es HTTP. Esto podría considerarse innecesario y una manera de complicar la comunicación entre aplicaciones cuando sería posible enviar los mensajes directamente sobre HTTP sin necesidad de SOAP.

De este pensamiento surge la idea de los servicios web basados en REST, cuya idea principal se basa en utilizar una interfaz web simple que utilice únicamente un conjunto de operaciones estándar como son las del protocolo HTTP para la transmisión de datos sin la necesidad de ningún otro protocolo basado en patrones de intercambio de mensajes (como es SOAP). Como con este tipo de servicios se trata de emular el modo de intercambiar datos utilizado por el protocolo HTTP, estos se centran más en interactuar con recursos que con mensajes y operaciones como sucedía con SOAP.

Un recurso es cualquier elemento de información en la red que represente un concepto de negocio y que posea un URI, es decir un identificador uniforme de recurso. Para manipularlos, clientes y servidores se comunican a través de una interfaz estándar (HTTP) e intercambian representaciones de los mismos. Una representación de un recurso es una copia del estado público del recurso en un formato concreto. Los recursos

pueden tener cero o más representaciones. Un ejemplo de recurso con su URI podría ser el siguiente:

http://www.futbol.com/equipos/RealMadrid

Este recurso podría ser una colección de elementos que a su vez, fueran recursos con sus respectivas URIs y por ejemplo, dos representaciones, una HTML y otra jpeg.

http://www.futbol.com/equipos/RealMadrid/Iker Casillas.
http://www.futbol.com/equipos/RealMadrid/Sergio Ramos.
http://www.futbol.com/equipos/RealMadrid/Cristiano Ronaldo..

Realmente REST no es un estándar, es un estilo de arquitectura de software para sistemas hipermedias distribuidos tales como la Web que hace uso de estándares:

- HTTP
- URL
- XML, HTML, JPEG, PNG, GIF etc. para las representaciones de los recursos.
- Tipos MIME: text/xml, text/html, imange/png, etc.

Un caso de arquitectura basada en REST que funciona a la perfección es la *world wide web*, o simplemente web. La motivación para crear servicios web basados en REST es la de emular a la propia web, utilizando sus principales características que hacen de la misma un éxito. A continuación se describirán los objetivos que se desean extraer de la web y las restricciones que se aplican en REST para alcanzarlos.

Objetivos y restricciones de REST

Los principales objetivos de la web que se desean alcanzar en una arquitectura de software REST son:

- *La escalabilidad en la interacción entre componentes* (capacidad de un sistema para trabajar con diferentes cantidades de trabajo). La web ha crecido mucho desde su puesta en marcha y seguirá creciendo en el futuro. Todo ello sin empeorar su rendimiento incluso teniendo en cuenta la gran variedad de clientes que han aparecido con el tiempo: estaciones de trabajo, sistemas industriales, dispositivos móviles, etc.
- *La generalidad de la interfaz*. Gracias a que la web utiliza el protocolo de transporte HTTP cualquier cliente puede interactuar con cualquier servidor sin ninguna configuración especial, simplemente usando sus operaciones definidas.
- *La extensibilidad del sistema*. (capacidad de un sistema para incluir mecanismos para la expansión/mejora del mismo con capacidades previstas sin tener que hacer cambios importantes en su infraestructura.) En la web está en constante crecimiento. Clientes y servidores están en funcionamiento durante años y es necesario que éstos sean capaces de interactuar con clientes y servidores más modernos y viceversa. La web lo consigue mediante el protocolo HTTP mediante el uso de cabeceras (cabeceras especiales

como *Accept* o *Content-Type* pueden especificar que representaciones entienden el servidor y el cliente y que representación se usa en un mensaje concreto para transportar el estado del recurso.), a través de las URIs o mediante la habilidad para crear nuevos tipos de contenido.

- *Compatibilidad con componentes intermedios*. La web posee una infraestructura propia de componentes intermedios. La compatibilidad con componentes tales como cachés, firewalls y puertas de enlace (dispositivos que permite interconectar redes de computadoras con protocolos y arquitecturas diferentes a todos los niveles de comunicación.) ofrece una serie de ventajas como la mejora en rendimiento y seguridad y la posibilidad de encapsulamiento de sistemas.

Para poder conseguir los objetivos comentados, REST define un conjunto de restricciones o características que se deben seguir en el desarrollo de este tipo de servicios web:

- *Uso de una interfaz uniforme*. En REST los servicios no publican una serie de métodos u operaciones concretas para cada servicio como ocurría en los servicios basados en SOAP. Como se comento anteriormente, REST se centra en recursos y para acceder a éstos existe una interfaz única y constante. Todos los recursos comparten las mismas operaciones. Las operaciones permiten manipular el estado público del recurso. En un sistema REST típico se define una interfaz con cuatro operaciones.

    - **CREATE.** Mediante esta operación un cliente manda al servidor una petición para crear un nuevo recurso. Opcionalmente el cliente puede mandar una representación del estado inicial de este recurso. El servidor responde con el URI del nuevo recurso.
    - **DELETE.** Con esta operación se envía una petición para eliminar un recurso del servidor. El usuario necesita saber el URI del recurso y tener permiso para hacerlo.
    - **READ.** Con esta operación el cliente envía una petición para obtener una representación del estado de un recurso, identificado con su URI. El cliente puede especificar qué tipos de representaciones entiende. Mediante esta operación se realiza una copia el estado del recurso en el servidor y se pega en el cliente. La copia del cliente no se mantiene sincronizada con el recurso en el servidor. El servidor puede cambiar el estado real del recurso y el cliente, de forma independiente, puede modificar su copia local del estado del recurso.
    - **UPDATE.** Como el servidor y el cliente tienen una copia diferente del estado, el cliente puede usar esta operación para sobrescribir o grabar su copia del estado en el servidor. De esta manera se puede actualizar el estado del recurso con las modificaciones hechas en el cliente.

Una de las características claves de los servicios web REST es el uso explícito de los métodos HTTP para ejecutar estas operaciones. Por ello se establece una asociación uno-a-uno entre las operaciones *create, delete, read y update* y los métodos HTTP. De acuerdo a esta asociación:

- se usa POST para realizar la operación *create*.
- se usa GET para realizar la operación *read.*
- se usa PUT para realizar la operación *update.*
- se usa DELETE para realizar la operación *delete.*

- *Sin mantenimiento de estado*. Los servicios web basados en REST deben ser stateless que significa sin estado. Esto quiere decir que entre dos solicitudes cualesquiera entre un cliente y el servicio, este último no conserva ningún dato sobre el cliente. Para el servicio la segunda petición es una petición completamente nueva, no guarda sesiones de usuario. Es el propio usuario el que debe mantener el estado y enviarlo al servidor en cada solicitud. Esta forma de funcionar puede resultar tediosa pero tiene una gran ventaja, la escalabilidad. Para mantener el estado de un usuario hace falta memoria donde almacenarlo, en caso de que haya muchos usuarios el servidor podría quedarse sin memoria, por lo tanto podría llegar el momento en que no se pudieran admitir más clientes. Esto se podría solucionar utilizando varios servidores o bases de datos donde almacenar los estados pero son soluciones muy ineficientes que influyen mucho sobre el rendimiento, no recomendables si se desea tener un sistema altamente escalable.

- *Sintaxis universal para definir los recursos.* En un sistema basado en REST cada recurso debe ser únicamente direccionable a través de su URI. Como se comentó anteriormente, un URI es un identificador único de recurso que distingue un recurso de cualquier otro. Los URIs de los recursos de un servicio web basado en REST deben ser intuitivas y fáciles de adivinar de forma que un usuario sean capaz de entender a lo que apunta y los recursos derivados de éste. La forma de lograrlo es definir los URIs como una estructura al estilo de directorios, es decir una estructura jerárquica con un URI raíz y extendiendo subrutas de recursos.

- *Recursos con múltiples representaciones.* Cada recurso posee un estado interno que no es accesible por los usuarios del servicio. A lo que acceden los usuarios es a las diferentes representaciones posibles de dicho recurso. Una representación es una copia del estado interno de un recurso en un formato concreto que se transfiere entre el usuario y el servicio. Es el desarrollador del servicio el que define que tipos de representaciones son accesibles por los usuarios. La representaciones pueden ser en cualquier formato imaginable, tanto datos estructurados (documentos XML, HTML o JSON), como otro tipo de formatos tipo PNG, GIF, PDF, documentos Excel, etc. Las representaciones se definen mediante los atributos *Accept* y *Content-Type* de HTTP y tipos MIME. La mayoría de los tipos MIME son estándares, como XML o JSON. Como ya se vio en SOAP el usar protocolos y ,en este caso tipos MIME estándar, facilita la interoperabilidad.

- *Componentes en capas:* Intermediarios, tales como firewalls, servidores caché, puertas de enlace, etc., pueden ser introducidos entre los clientes y los recursos, para ofrecer mejor rendimiento, seguridad, y mejorar la escalabilidad, ya que permiten mover funcionalidades de uso infrecuente hacia componentes intermedios mejorando el balanceo de carga.

¿Cómo funcionan?

A continuación se mostrará un ejemplo del funcionamiento de un servicio web basado en REST. En el ejemplo, la empresa ficticia de venta de libros llamada compratuslibros pone a disposición de sus clientes un servicio web REST que les permite obtener una lista de libros, obtener información detallada sobre un libro en particular y enviar una orden de compra.

Imaginemos que un cliente desea acceder a la lista de libros disponibles. En esta ocasión no existen operaciones del tipo getListadoLibros() como ocurría al crear un cliente a partir de un descriptor WSDL de un servicio SOAP. En este caso se accede mediante identificadores URI a los recursos disponibles directamente mediante la interfaz uniforme.

Para acceder al listado de libros se accede al recurso que será un listado de libros mediante su URI con el método de HTTP GET que como comentamos anteriormente envía una petición para recibir una representación del recurso seleccionado. Quedaría de la siguiente manera:

```
GET https://www.compratuslibros.com/libros
```

La generación del listado de libros disponibles es totalmente transparente al cliente. Todo lo que conoce es que ha solicitado el listado a través del URI y que el documento que contiene el mismo le ha sido devuelto. El desarrollador del servicio es libre de modificar la estructura de recursos que son accesibles sobre el servicio web sin que el cliente deba realizar ningún cambio. Seguirá utilizando la misma interfaz sencilla sobre la nueva estructura de recursos. De esta forma se consigue bajo acoplamiento entre las aplicaciones clientes y los servicios. El listado que sería devuelto sería como este:

```
<?xml version="1.0"?>
<l:Libros xmlns:l="https://www.compratuslibros.com"
xmlns:xlink="http://www.w3.org/1999/xlink">
     <Libro nombre="Don Quijote de la Mancha"
     xlink:href="https://www.compratuslibros.com/libros/Don
Quijote de la Mancha"/>
     <Libro nombre="1984"
     xlink:href=" https://www.compratuslibros.com/libros/1984"/>
     <Libro nombre="2001 una odisea espacial"
     xlink:href=" https://www.compratuslibros.com/libros/2001_una
odisea espacial"/>
     <Libro nombre="Juego de Tronos"
     xlink:href=" https://www.compratuslibros.com/libros/Juego de
tronos"/>
</l:Libros>
```

El servicio ha devuelto a la petición del cliente del recurso *listado de libros* una representación del mismo en formato XML. Como se observa cada libro en el listado posee el URI del recurso concreto, es decir de un libro específico. Como se comento en un apartado anterior los URI de los recursos deben ser fáciles de seguir y adivinar. En este caso mediante el listado el cliente ya puede conocer el URI de los recursos que componen el listado, por lo que podrían hacer una petición sobre cualquiera de ellos fácilmente.

Ahora el usuario desea conocer la información de uno de los libros. Mediante el siguiente comando accede a su URI y recibe el resultado, , una representación del libro elegido en formato XML.

```
GET https://www.compratuslibros.com/libros/1984
```

```
<?xml version="1.0"?>
<l:Libro xmlns:l="https://www.compratuslibros.com"
xmlns:xlink="http://www.w3.org/1999/xlink">
<ISBN>9788499890944</ISBN>
<Nombre>1984</Nombre>
<Autor>George Orwell</Autor>
<Editorial>Debolsillo</Editorial>
<Portada
xlink:href="https://www.compratuslibros.com/libros/1984/portada"/
>
<Lengua>Castellano</Lengua>
<Moneda>Euro</Moneda>
<Cantidad>7,55</Cantidad>
</l:Libro>
```

Con esta petición se devuelve la información de un libro en concreto. Como se puede observar sigue siendo sencillo encontrar los URIs. En este caso el usuario podría hacer la petición para obtener una representación de la portada del libro en cualquier otro formato de imágenes que estuviera definido en el servicio.

A continuación imaginemos que un usuario desea comprar el libro sobre el cual ha estado informándose. El usuario debe crear un documento de instancia de orden de compra ajustándose al esquema definido por la empresa y lo envía mediante la operación POST. El usuario tiene que enviar también su identificación porque como se explicó anteriormente el servicio no conserva el estado de la comunicación:

```
POST https://www.compratuslibros.com/ordenesCompra/
/Orden.xml?id_usuario=1544
```

El servicio devolverá al usuario un URI donde el cliente puede encontrar la información de la orden en cualquier momento y así editarla o actualizarla mediante la operación PUT.

Por último la empresa desea eliminar un recurso del sistema porque va a dejar de vender uno de los libros. Usando la operación DELETE sobre el URI del recurso deseado se elimina del sistema.

```
DELETE https://www.compratuslibros.com/libros/1984
```

En el ejemplo comentado se accede a los recursos de forma directa mediante las operaciones de la interfaz simple de HTTP (GET, POST, PUT y DELETE), En un cliente programado en un lenguaje de alto nivel como Java podría se podrían utilizar librerías internas de Java como *java.net.HttpURLConnection* o *javax.net.ssl.HttpsURLConnection* para invocar las llamadas. Pero cuando se debe acceder a una gran cantidad de recursos esta manera puede resultar algo engorrosa. Por eso existen diferentes librerías que facilitan la integración de los clientes con las APIs REST de los proveedores. Algunas de ellas son *Jersey*, *RESTEasy* y *Restlet*. También existen otras librerías más centradas en APIs concretas como puede ser *Twitter4J* para las APIs REST de Twitter.

Toda la información explicada sobre servicios web basados en REST ha sido previamente estudiada de las siguientes fuentes [16] [17] [23].

2.5.2.4 Comparación entre servicios web basados en SOAP y en REST

Después de haber comentado ambos estilos de servicios web se podría preguntar porque es mejor usar los basados en SOAP que los basados en REST o viceversa. Ambos estilos tienen sus ventajas y desventajas. En este debate los desarrolladores web suelen posicionarse en uno o en otro bando. Unos comentan que SOAP es demasiado complicado y muy ineficiente ya que envían sus datos sobre dos protocolos de transporte SOAP Y HTTP, mientras que los otros opinan que SOAP es más flexible que REST a la hora de implementar servicios web (ya que pueden implementar las interfaces como deseen y no basarse en una fija), además que al ser fuertemente acoplados estos pueden ser fácilmente depurados y testados antes de ponerlos en funcionamiento, cosa que los primeros ven como una desventaja, prefieren REST por ser débilmente acoplados ya que en SOAP si se modifica un servicio y cambia su WSDL sería necesario modificar todos los clientes que lo consumen.

Este debate es difícil de cerrar, ya que ambos estilos tienen diferentes ámbitos donde funcionan mejor que el otro. Los Servicios web basados en SOAP serán mejores en arquitecturas que aborden requerimientos complejos no funcionales, como en casos donde vayan a existir transacciones donde sean necesarios seguridad o direccionamiento, casos donde es necesario mantener la información contextual y el estado. También cuando no se conocen el contexto y el contenido a intercambiar donde es necesario establecer un contrato que describa todos los detalles. Sin embargo REST es mejor cuando no se necesita mantener estados, cuando se utilizan sobre dispositivos con pocos recursos, en momentos donde el proveedor y el consumidor conocen el contexto y el contenido a intercambiar y en casos donde se necesite alto rendimiento ya que es posible usar componentes intermedios como cachés para evitar algunas consultas al servicio.

Para terminar con los servicios web, en las siguientes tablas de describirán a modo de resumen las características principales de cada estilo así como sus ventajas y desventajas.

	SOAP	REST
Características	• Estándar W3C de comunicación remota que utiliza el protocolo SOAP y estándares abiertos, lo que conlleva una alta compatibilidad con clientes independientemente del lenguaje y plataforma. • Las operaciones se definen mediante puertos en el descriptor WSDL. • Ofrecen una dirección única para todas las operaciones del servicio. • Múltiples instancias del proceso comparten la misma operación. (Por ejemplo una operación getLibros() obtendría en la misma operación todas las instancias de libro, no se puede acceder a un único libro). • Componentes fuertemente acoplados. Cuando se desarrolla un servicio, se desarrolla para que funcione como un todo, no se pueden extraer partes del mismo para que funcionen en solitario.	• Estilo de arquitectura para sistemas hipermedias distribuidos tales como la Web que se basan en estándares genéricos, lo que permite el acceso a cualquier tipo de cliente que se base en ellos. • Las operaciones se describen en los mensajes, es por esto que se llaman mensajes autodescriptivos. En el propio mensaje se indica qué operación realizar sobre qué recurso. • El servicio se compone de un conjunto de recursos. Cada uno de ellos con una dirección única. • Cada recurso del servicio soporta las operaciones estándares definidas. • Componentes débilmente acoplados. Los recursos como tal son independientes unos de otros.

Tabla 1. Características principales SOAP Y REST.

	SOAP	REST
Ventajas	• Facilidades en su utilización, ya que existen mecanismos que permiten generar las interfaces del servicio a partir de su WSDL y también generar el WSDL a partir de la lógica del negocio. • La depuración es posible. • Las operaciones pueden ser escondidas detrás de una fachada. Todos los detalles de implementación se ocultan al cliente. Para accederlo se centra en lo definido en los WSDL sin importar cómo funciona. • Incrementa la privacidad, ya que sólo se accede contra la fachada de servicio, sin conocer la estructura interna del mismo. • En caso de necesidad de utilizar otro protocolo de transporte diferente de HTTP (SMTP o FTP) SOAP lo acepta.	• Bajo consumo de recursos. No es necesario un tratamiento extra de información como ocurre en SOAP con el envelope. • Escalabilidad: Cada vez hay más sistemas capaces de acceder a los servicios y estos no pierden rendimiento. • Las instancias del proceso son creadas explícitamente. • Los clientes no necesitan información de enrutamiento a partir de la URI inicial. Los recursos son autodescubribles sin necesidad de un documento descriptor. • Generalmente fáciles de construir y adoptar. • Aprovecha la infraestructura de la web ya existente (cachés, proxies, firewalls). • Soporta muchos tipos de representaciones para un mismo recurso (xml, pdf, png, gif, excel, html, json) de forma natural.
Desventajas	• Los clientes necesitan saber las operaciones y su semántica antes del uso. No existen operaciones genéricas como en REST con las que invocar directamente a recursos. • Las instancias del proceso son creadas implícitamente. • Mayor consumo de recursos ya que necesita intercambiar mensajes SOAP sobre HTTP, lo que influye en el rendimiento.	• Normalmente tienen un gran número de recursos, cada uno con su URI. Como se accede a cada recurso por separado, un espacio de nombres grande puede llegar a ser complicado de manejar. • La descripción sintáctica/semántica es muy informal, orientada al usuario. • Menos flexible que SOAP en cuanto a la creación de interfaces. Estos se basan en una interfaz fija. • En caso de necesidad de utilizar otro protocolo de transporte diferente de HTTP (SMTP o FTP) REST no puede funcionar.

Tabla 2. Ventajas y desventajas SOAP y REST.

2.5.3 RSS

¿En qué consiste?

RSS [13] son las siglas de Really Simple Syndication. Es un sencillo formato de datos definido en la versión 1.0 de XML, que es utilizado para difundir las novedades en los contenidos de un sitio web a los suscriptores del mismo. Este formato permite difundir información sin la necesidad de que los usuarios accedan por medio del navegador a la información de las diferentes páginas web de su interés. Utilizando un software diseñado para leer estos contenidos RSS (llamados agregadores o lectores de RSS) los usuarios pueden recibir alertas sobre las novedades en las páginas web de su interés a las que estén suscritos de forma centralizada, sin tener que visitar cada página web por separado. Las versiones actuales de los principales navegadores también permiten leer los RSS sin necesidad de software adicional.

Cualquier usuario puede suscribirse a uno o varios canales RSS de los sitios web de su interés que dispongan del servicio RSS y obtener las últimas novedades en su lector RSS, el cual le alertará cuando haya nueva información para leer en cualquiera de los canales (también llamados Feeds) a los que se haya suscrito. Esto agiliza la forma de mantenerse informado y ahorra tiempo, pues no es necesario comprobar los múltiples sitios web sin saber si se habrá producido algún cambio en ellos o no. Los agregadores de feeds no ofrecen la información completa, simplemente un resumen de ésta. Tras recibir la alerta del lector RSS, si se desea ampliar el contenido será necesario visitar el sitio que ha originado la información siguiendo el enlace proporcionado con la alerta.

Pero RSS no sólo sirve para recibir la información que otros le ofrecen, sino que cualquiera que posea una página web o un blog puede usar este formato para difundir su contenido a otros internautas. Para ello se necesita que el usuario cree su propio canal RSS y lo actualice frecuentemente con la última información. Esta forma de difundir la información puede ayudar a aumentar el tráfico de usuarios en un sitio web ya que de esta forma no se espera por la visita de los usuarios sino que se contacta con ellos para mostrarles información que pueda atraerlos al sitio.

Formato de un documento RSS

La especificación utilizada actualmente para la creación de los RSS es la 2.0 [24]. Esta especificación describe las etiquetas utilizadas en la creación de un canal RSS.

El formato de un documento RSS esta descrito mediante XML. Lo primero que encontramos en un documento RSS son las líneas que declaran que el fichero es XML, el tipo de datos a utilizar y posteriormente la versión RSS utilizada.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rss version="2.0">
</rss>
```

El resto de componentes que conforman el fichero RSS las encontramos entre las etiquetas <rss>. Lo primero que encontramos dentro de ellas es la etiqueta <channel> que define un "canal" en el que se encuentran los contenidos que se van mostrar. La etiqueta <channel> está compuesta de otros elementos obligatorios y opcionales. Los elementos obligatorios de un canal son:

- el título (<title>), que hace referencia al nombre del feed o canal.
- el enlace (<link>), que es la URL del sitio Web al que pertenece el canal.
- la descripción (<description>), que informa al usuario del tipo de contenidos que se incluyen en el canal.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rss version="2.0">
 <channel>
  <title>El nombre de nuestro feed</title>
  <link>Dirección web en la que se encuentre nuestro RSS</link>
  <description>Contenido que vas a ofrecer a los usuarios</description>
 </channel>
</rss>
```

Y los opcionales que como su propio nombre indica no siempre aparecen dentro de la etiqueta <channel>:

- <language>, especifica el lenguaje en el que está escrito el canal.
- <copyright>, la notificación del copyright para el contenido del canal.
- <managingEditor>, El correo electrónico del responsable del contenido del canal.
- <webMaster>, El correo electrónico de la persona responsable de los asuntos técnicos relacionados con el canal.
- <pubDate>, La fecha de publicación del contenido en el canal.
- <lastBuildDate>, la última vez que cambió el contenido del canal.
- <category>, especifica las categorías a las que el canal pertenece.
- <generator>, indica el programa usado para generar el canal.
- <docs>, una URL que apunta a la documentación del formato utilizado en el RSS.

- <ttl>, significa tiempo de vida. Se trata del número de minutos que un canal puede almacenar en caché antes de actualizarse desde la fuente.
- <image>, especifica una imagen GIF, JPEG o un PNG que pueden ser desplegados con el canal. Esta imagen estará definida por:
  - <url>: La URL de la imagen.
  - <title>: El título que describe la imagen.
  - <link>: La URL del sitio web al que pertenece.
- <skipHours>, indica las horas del día que no se podrá acceder al canal.
- <skipDays>, indica los días de la semana que no se podrá acceder al canal.

Además de estos atributos que describen el canal, dentro de la etiqueta <channel> también se encuentran lo elementos <item> que representan la información que se difunde a través del canal. Esta etiqueta está repetida tantas veces como elementos de información se vayan a difundir. Todos los elementos de un <item> son opcionales, sin embargo al menos el título, la descripción y el link deben aparecer:

- <title>: El titulo del ítem.
- <description>: Una sinopsis del ítem en texto plano.
- <link>: Dirección Web a la que podemos ir para ampliar esta información.
- <author>: El correo electrónico del autor del ítem.
- <category>: Especifica las categorías a las que pertenece el ítem.
- <content:encoded>: Incluye una descripción en formato HTML.
- <comments>: URL de una página de comentarios relacionados con el ítem.
- <enclosure>: Describe el objeto multimedia que se adjunta al ítem.
- <guid>: Una cadena que identifica unívocamente el ítem.
- <pubDate>: Indica la fecha de cuando fue publicado el ítem.
- <source>: El canal RSS del cual viene el ítem.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rss version="2.0">
 <channel>
  <title>El nombre de nuestro feed</title>
  <link>Dirección web en la que se encuentre nuestro RSS</link>
  <description>Contenido que vas a ofrecer a los usuarios</description>
  <item>
   <title>Título del artículo</title>
   <link>Dirección Web a la que podemos ir para ampliar
       esta información</link>
   <description>Contenido de esta información</description>
  </item>
 </channel>
</rss>
```

¿Cómo se consume un canal RSS?

Se acaba de describir la estructura completa de un RSS 2.0 y con ella, cualquier agregador o lector RSS podrá acceder a la información y disponerla para que la vean los usuarios.

Existen varios tipos de lectores RSS disponibles a través de internet. Podemos encontrar lectores que se instalan en el ordenador y lectores RSS online que se acceden a través del navegador. Ambos tipos se encargan de mostrar los elementos de los canales RSS de los cuales se hayan indicado su dirección, ellos mismos se encargan de procesar los elementos y de visualizarlos.

Pero aparte de los lectores de RSS que ya están listos para usar, existen soluciones para poder crear tu propio lector donde se puede procesar la información y visualizarla de la forma deseada. Una muy buena solución para crear un lector de RSS propio es la librería de Google, Google Feed API.

Google Feed API permite descargar la información de los canales RSS usando simplemente lenguaje JavaScript. Una vez obtenida la información (mediante objetos JSON, en XML o con una combinación de ambos según se configure cuando se utiliza la API), ésta puede ser procesada para obtener los elementos deseados y disponerlos de la forma que deseemos y mostrarlos a los usuarios. La ventaja de usar este API es que todo el procesamiento pesado y todo el trabajo complejo (como lidiar con proxies en el lado del servidor), lo realizan los servidores de Google de una forma óptima, permitiendo manejar el resultado de forma sencilla, con unas pocas líneas de código JavaScript.

Esta es la librería que se utilizará para desarrollar el lector de RSS que formará parte del portal de sindicación de información objetivo del proyecto. Toda la información referente a esta librería se puede encontrar en su página oficial [25].

2.5.4 EJB

Enterprise Java Beans (EJB) es una plataforma para construir aplicaciones de negocio portables, reusables y escalables usando el lenguaje de programación Java. [26] Desde el punto de vista de un desarrollador, un EJB puede entenderse como un componente del lado del servidor que encapsula la lógica del negocio de una aplicación. Este componente se ejecuta en un contenedor EJB, que no es más que un entorno de ejecución especializado que provee determinados componentes de servicio. Estos permiten al desarrollador centrarse en su totalidad en el desarrollo de la lógica de negocio ya que se encargan de manejar la mayoría de los problemas generales de una aplicación empresarial (concurrencia, transacciones, persistencia, seguridad, etc.)

Los servicios que debe proveer el contenedor de EJBs deben ser especificados por el programador a través de metadatos de configuración que puede escribirse como descriptores XML (archivos XML separados) o mediante anotaciones de Java (incorporadas a partir de Java 5 y disponibles en EJB desde la especificación 3.0) intercaladas en el código de las clases.

Como se ha comentado, la lógica del negocio reside en los enterprise beans y no en el lado del cliente. Esto permite que el desarrollo del lado del cliente esté desacoplado de la lógica del negocio. Los clientes acceden a través de una fachada contra el contenedor EJB que se encarga de gestionar el acceso, de este modo el acceso es transparente para el cliente.

Existen diferentes tipos de EJB pero aquí se describirán únicamente los estudiados para su posterior uso en el proyecto: Los EJB de sesión (en inglés session beans).

Los EJB de sesión forman parte de la capa de la lógica del negocio en una aplicación empresarial típica y son invocados por el cliente con el propósito de ejecutar operaciones de negocio específicas. El término sesión implica que una instancia del bean está disponible durante una unidad de trabajo y no sobrevive a más allá de la misma ni a , por ejemplo, caídas del servidor.

Los beans de sesión están compuestos por una o más interfaces y una clase de implementación. Un cliente puede acceder a un session bean solamente a través de métodos definidos en las interfaces del bean. Estas interfaces pueden ser de dos tipos: locales o remotas. Los clientes remotos (un componente web, una aplicación cliente u otro enterprise bean) son aquellos que se ejecutan en una JVM (máquina virtual java) diferente a la del Enterprise bean al que invocan y accederán a través de la interfaz remota. Los clientes locales (un componente web u otro enterprise bean) son aquellos que se ejecutan en la misma JVM que la del Enterprise bean al que invocan y accederán a través de la interfaz local.

Una aplicación EJB normalmente se empaqueta en un archivo EAR conteniendo la clase de implementación y ambas interfaces para su posterior despliegue. Esto puede llevar a confusión en cuanto a entender cuando se accede remotamente y localmente. Se tiende a pensar que cuando el cliente del EJB y la propia aplicación EJB se ejecutan sobre el mismo servidor de aplicaciones ambos están corriendo sobre la misma máquina virtual de java por lo que se debería usar la interfaz local, pero no siempre ocurre así, depende del servidor de aplicaciones. La especificación EJB3 únicamente obliga a que el interface de acceso local esté disponible para los clientes que se encuentran ubicados en el mismo EAR que la aplicación EJB. Es decir, que si la aplicación cliente no se encuentra empaquetada en el mismo EAR que la aplicación EJB, dependiendo del servidor de aplicaciones utilizado, el cliente deberá utilizar la interfaz local o por el contrario la remota. Si se quiere asegurar que el acceso funciona en cualquier servidor de aplicaciones se debe construir un EJB con acceso remoto [27].

Ilustración 7. Arquitectura EJB.

Hay dos tipos de Session Beans: Stateless y Stateful:

- Stateless (sin estado). En los beans de sesión sin estado, Las instancias son compartidas por los clientes. Las variables de la instancia mantienen estado específico del cliente, pero sólo por la duración de la invocación de un método. El contenedor del EJB tiene un conjunto (o pool) de instancias, cuando el cliente invoca un método se asigna una instancia, cuando se termina la ejecución y la libera es retornada al conjunto. Estos beans de sesión al no mantener el estado permiten que se los acceda concurrentemente y ofrecen mejor escalabilidad para aplicaciones con gran cantidad de clientes.

- Stateful (con estado). En un bean de sesión con estado, una instancia del bean se reserva para el cliente. Las variables de la instancia reservada almacenan datos específicos obtenidos durante la conexión con el cliente. Cada bean de sesión con estado, por tanto, almacena el estado conversacional de un cliente que interactúa con el bean. Este estado conversacional se modifica conforme el cliente va realizando llamadas a los métodos de negocio del bean. El estado conversacional no se guarda cuando el cliente termina la sesión.

Como ya se ha comentado, cuando un cliente del EJB desea acceder a sus métodos debe acceder a través de alguna de sus interfaces. Para ello necesita localizarla y obtenerla. Para poder obtener la interfaz, se usa el llamado servicio de integración, el cual provee el contenedor de EJB. Este servicio ofrece una forma de acoplar en tiempo de ejecución diferentes componentes, mediante una simple configuración de anotaciones. Existen dos formas de realizar este acoplamiento: mediante inyección de dependencia (sólo disponible a partir de EJB 3.0) o usando JNDI lookup.

La inyección de dependencia "inyecta" objetos en una clase, en lugar de ser la propia clase quien cree el objeto. Por ejemplo, podría insertar un objeto interfaz del EJB en la clase de una aplicación cliente sin que ésta conozca la implementación de la interfaz. Realmente la inyección es una orden directa al contenedor de EJBs para que cree una instancia de tipo interfaz con la implementación proporcionada del lado del servidor.

JNDI es un servicio de nombres que permite a un componente localizar otros componentes o recursos. Cuando se usa JNDI, es responsabilidad del cliente buscar y obtener la referencia al objeto. La inyección es exactamente lo opuesto a JNDI: es responsabilidad del contenedor inyectar un objeto basado en una dependencia declarada.

En el caso particular del proyecto a desarrollar se van a crear varios portlets que serán clientes de un mismo servicio web. Cuando se crea un cliente de un servicio web en Java a partir de su descriptor WSDL se genera un gran número de clases que permiten consumirlo. Esto es un problema ya que al crear varios clientes del mismo servicio se generan todas estas clases en cada portlet, lo cual es muy ineficiente ya que se repite muchísima cantidad de código. Para solucionarlo se va a utilizar los EJBs. Se generara una aplicación EJB que hará de cliente del servicio web y que será la única que poseerá las clases generadas desde el WSDL. A partir de ahí los portlets que van a consumir el servicio se conectarán al EJB a través de su interfaz que ofrecerá la información obtenida de servicio web.

2.5.5 AJAX

¿Qué es?

AJAX [19] es una tecnología de programación web que es casi imprescindible hoy en día. La mayoría de páginas web, por no decir todas usan la tecnología AJAX en alguna de sus partes. Aunque muy a menudo se nombra AJAX como una única tecnología, en realidad es el conjunto de otras tecnologías bien conocidas en la programación web trabajando conjuntamente. Como su propio nombre indica, AJAX (Asinchronous JavaScript and XML) está formada por JavaScript, y XML además de CSS. Estas tecnologías permiten hacer llamadas asíncronas al servidor donde reside el verdadero poder de AJAX.

¿Cómo funciona? Diferencia modelo clásico.

Con AJAX se cambia el modo de interactuar entre el cliente y el servidor en la programación de aplicaciones web. Su principal característica es la de eliminar las recargas completas de página ya que es posible recargar una parte especifica de esta. Además, por este mismo motivo se evita que el usuario pierda el contexto de lo que está haciendo ya que nunca perderá el control de la aplicación. A su vez se consiguen que los tiempos de respuesta de las peticiones al servidor se reduzcan al tener que gestionar y devolver mucha menos cantidad de información, ya que únicamente devolverá los datos de la parte a actualizar.

La diferencia con el modelo de web clásico reside en lo que puede llamarse el motor AJAX. Puede considerarse una capa intermedia entre el cliente y el servidor. Mientras en el modelo clásico las peticiones se envían desde el cliente directamente al servidor y se le pide que actualice la página con los nuevos datos (dejando al usuario esperando la respuesta), con AJAX las peticiones se envían mediante funciones JavaScript al motor AJAX que es el encargado de gestionar las peticiones al servidor de forma transparente y de devolver los datos cuando estén disponibles. En ese momento se podrán incluir esos nuevos datos devueltos en formato XML o texto plano, modificando el árbol DOM [28] de la interfaz. Este cambio se puede hacer en tiempo de ejecución sin tener que recargar la página completa por lo tanto el usuario no perderá el control de la aplicación en ningún momento.

¿Por qué usarlo?

Como he comentado anteriormente los portlets funcionan con peticiones síncronas, si se quiere cambiar de pantalla en un portlet concreto haciendo una petición al servidor, se tendrá que recargar toda la página del portal para únicamente actualizar uno de los portlets, por lo que utilizar AJAX es recomendable en algunos casos para evitar estas cargas completas del portal en las que el usuario pierde momentáneamente el control de la aplicación.

Librerías.

Existen gran cantidad de librerías JavaScript que ofrecen funciones AJAX y que facilitan mucho el desarrollo usando esta tecnología. En mi caso he utilizado JQuery debido a que es una de las librerías más conocidas, es fácil de usar y aprender. Además hay muchísima información oficial y extraoficial en internet. Esta librería se puede usar junto con JQuery User Interface para crear la interfaz de usuario. JQuery ofrece funciones para realizar peticiones AJAX, manejar y modificar el árbol DOM, controlar eventos de usuario y añadir diferentes efectos a la interfaz. Si se desea obtener más información sobre cómo funciona, la API de la página oficial [29] es un buen punto de inicio. Para profundizar en JQuery se ha estudiado también el libro online Fundamentos de JQuery [30]

2.6 Conclusión

Una vez visto el estado del arte y definidas las tecnologías que se van a utilizar en el desarrollo del proyecto, en el siguiente capítulo se va a describir el entorno de desarrollo, tanto la instalación cono la configuración del mismo. Además se describirán el hardware y las versiones específicas de las herramientas y tecnologías utilizadas en el desarrollo y la ejecución del proyecto.

Capítulo 3

Entorno de desarrollo

3.1 Introducción

Una vez detalladas las tecnologías que se van a usar para la creación de la aplicación, en este capítulo se van a exponer las características de Hardware y Software que se han utilizado para el estudio, realización y ejecución del proyecto. También contendrá un apartado de ayuda que indicará como se instala y configuran los diferentes elementos y tecnologías involucradas en el desarrollo del proyecto. En el caso de mi aplicación es muy importante realizar una configuración minuciosa, utilizando versiones concretas del software indicado para que todo funcione correctamente, si se utilizan otras versiones es posible que se produzcan errores de compatibilidad que eviten el correcto funcionamiento del entorno de desarrollo y por lo tanto de la aplicación.

3.2 Hardware

Tanto para el desarrollo como para la ejecución se ha utilizado el mismo hardware debido a que la aplicación se ejecuta en un entorno local.

Durante el proceso de desarrollo se han utilizado dos ordenadores diferentes. En un principio comencé a desarrollar con el siguiente hardware:

Fabricante: Toshiba

Modelo: Satellite A300

Procesador: Intel Core 2 Duo CPU T8100 @ 2.10GHZ 2.10GHz

Memoria principal: 3GB

Grafica: Intel 965 Express Chipset Family

Tras adquirir un nuevo ordenador, por motivos de rapidez y seguridad se hizo una migración del proyecto. Sobre este hardware es el que se realizará la presentación:

Fabricante: Mountain

Modelo: GTM 17 IVY

Procesador: Intel Core I7-3740QM CPU @2,70GHz 2,70GHz

Memoria principal: 32GB

Grafica: NVIDIA GTX 680M

Hardware adicional:

Ratón Bluetooth Logitech.

3.3 Software

En el siguiente apartado se exponen los diferentes elementos software de los que se compone el proyecto. Se incluye el software utilizado para el desarrollo, ejecución y creación de la memoria.

3.3.1 Sistema Operativo

Como ya he comentado para el desarrollo y ejecución de mi aplicación web se han utilizado dos ordenadores diferentes con dos sistemas operativos diferentes:

Windows Vista

El sistema operativo instalado en el primer ordenador es Windows Vista Home Premium.

Windows 7

Tras realizar la migración al nuevo ordenador la aplicación se ejecuta sobre Windows 7 Enterprise Edition de 64 bits.

3.3.2 IDE

NetBeans Versión 6.9.1

Para desarrollar aplicaciones web es necesario hacer uso de un IDE (entorno de desarrollo integrado), que facilita en gran medida la tarea del desarrollador. Como la aplicación va a ser desarrollada en Java es necesario elegir un IDE que permita desarrollar en este lenguaje. Buscando entre los más populares dentro del software libre se ha elegido Netbeans, hecho principalmente para el desarrollo con Java. El principal motivo para elegirlo es que además de estar pensado para desarrollar con Java y ser software libre, es posible extenderlo con gran cantidad de módulos y plugins cosa muy necesaria cuando se quieren programar portlets.

Existe gran cantidad de información, foros, y grupos de programadores que ayudan a resolver los diferentes problemas que puedan surgir a la hora de desarrollar. En el sitio web oficial se pueden obtener las diferentes versiones que se ofrecen. Si se desea, se puede elegir descargar NetBeans junto con el servidor de aplicaciones GlassFish.

En mi caso concreto he utilizado NetBeans Versión 6.9.1. Se puede descargar bajo licencia CDDL y GPL2 desde la página oficial. [31] Es esencial descargarse esta versión ya que si no lo hacemos obtendremos problemas de incompatibilidades con otros elementos software utilizados.

Portal Pack

Se ha añadido un plugin para Netbeans que añade una interfaz adicional, incluyendo nuevas posibilidades de creación, configuración y desarrollo de portlets, lo cual facilita la labor del programador y evita la inclusión de diferentes errores relacionados con estos.

El nombre del plugin en cuestión es Portal Pack v3.0.4. Se puede descargar desde la página de NetBeans [32] en el apartado dedicado a este módulo. Esta es una de las últimas versiones del plugin, la cual he utilizado para desarrollar los diferentes portlets y está optimizada para el uso con NetBeans 6.9.1 de ahí la importancia de descargar la versión correcta.

3.3.3 JDK

Como Liferay es un portal basado en Java y voy a desarrollar los portlets utilizando el lenguaje Java es necesario instalar un JDK (Java Development Kit).

Para desarrollar en Java es necesario descargar e instalar un JDK, que ofrece el conjunto de herramientas y librerías necesarias para desarrollar con este lenguaje y ejecutar las aplicaciones creadas. Hay que tener en cuenta que descargarse únicamente el JRE (Java Runtime Environment) no será suficiente ya que este sólo ofrece un conjunto de utilidades para ejecutar aplicaciones Java, no para desarrollarlas. El JRE viene incluido en el JDK. Desde la página de Oracle [33] se puede acceder a la página de descargas donde podremos elegir la versión que se desee. La versión más actual en el momento de escribir esta líneas el JDK 8 u.40. En mi caso he utilizado el JDK 6 u.23 debido a motivos de compatibilidad con el resto de componentes.

3.3.4 Servidores

Para la creación y ejecución del proyecto se han utilizado 3 servidores diferentes. Un servidor de aplicaciones que se encargara de ejecutar la aplicación, incluido el portal Liferay. Un sistema gestor de base de datos que es necesario para almacenar los datos de configuración para el correcto funcionamiento del portal Liferay. También almacenará parte de la información recogida de las fuentes web utilizada por alguno de los componentes desarrollados. Por último, el servidor Liferay que, aunque se puede considerar un servidor virtual, es más una aplicación especial que corre sobre el servidor de aplicaciones y que se encarga de todo lo relacionado con la gestión del portal.

GlassFish Server

Es un servidor de aplicaciones de código libre desarrollado en Java por Sun Microsystems. Sobre él se ejecutará la aplicación web objetivo de este proyecto. En este caso, al tratarse de un portal web, este servidor es el encargado de soportar y ejecutar Liferay.

La versión utilizada en el proyecto es GlassFish 3.0.1 .El sitio web oficial ofrece información variada y un apartado de descargas de donde se puede conseguir este servidor bajo licencia gratuita CDDL y GPL [34].

Liferay Portal

Liferay es un portal web sindicador de contenido de código abierto escrito en Java. Como comenté en el apartado del estado del arte, es una aplicación web adaptable por el usuario en cuanto a contenido y estructura, que ofrece acceso centralizado a una gran variedad de contenidos y aplicaciones utilizando información proveniente de diferentes fuentes web.

He considerado diferenciar Liferay como un servidor virtual, pues así lo define el propio NetBeans una vez desplegado, sin embargo no es más que una aplicación WAR

que corre sobre un servidor de aplicaciones, en mi caso concreto GlassFish, aunque puede ser ejecutado sobre la mayoría de ellos.

La versión utilizada de Liferay es Liferay 6.0.5. Se puede obtener de la página oficial bajo licencia LGPL . [11]

MySQL Server

Para poder instalar y ejecutar el portal Liferay es necesaria una base de datos donde almacenar los datos de configuración, por lo tanto será necesario tener instalado un sistema gestor de base de datos. En este caso he elegido MySQL. Es un sistema gestor de base de datos relacional de código libre muy fácil de instalar y utilizar. En la página web oficial se puede descargar la última versión disponible, así como el Workbench que proporciona a los diseñadores y desarrolladores un entorno de herramientas integradas en modo gráfico para administrar, diseñar y modelar base de datos y para desarrollar sentencias SQL. También es necesario un driver de conexión para Java que permitirá conectarse desde la aplicación, a las bases de datos creadas. Este driver viene incluido en la descarga de MySQL Server.

Las versiones que he utilizado en el desarrollo de mi aplicación son MySQL Community Server 5.6.21 y MySQL Workbench 6.2, ambas descargables bajo licencia GPL, desde el apartado downloads del sitio web oficial [35].

3.3.5 Librerías

Para el desarrollo de los diferentes portlets que componen la página principal del portal se han utilizado diversas librerías que facilitan mucho la tarea de programar. La mayor ventaja de utilizar este tipo de librerías es que ahorra muchos problemas y tiempo, disminuyendo en gran medida la dificultad y por lo tanto el número de posibles errores a la hora de programar.

Para cada portlet basado en una tecnología, se ha utilizado la librería correspondiente que ayuda a programar sobre la tecnología en cuestión. Cada una de ellas tiene un objetivo concreto. Las librerías utilizadas se listan a continuación, indicando las versiones utilizadas.

Twitter4J

Twitter4J es una librería Java no oficial para la API REST de Twitter. Proporciona funciones de más alto nivel para las diferentes opciones que ofrece Twitter a los desarrolladores en cuanto a manejar y gestionar usuarios, mensajes, contenido multimedia, etc.

La versión utilizada es Twitter4J 3.0.3 que se puede descargar desde la página oficial [36] de la librería bajo licencia gratuita Apache License 2.0.

Google Feeds API

Google Feeds es una librería JavaScript de Google que permite descargar cualquier Atom público o fuente de medios RSS usando únicamente JavaScript. Esta librería facilita el consumo de este tipo de fuentes de información ya que se puede realizar mediante JavaScript en el lado cliente, por lo que se facilita la integración de este tipo de fuente de información dentro de un sitio web.

En la página oficial de la librería [25] hay gran cantidad de información sobre ella, incluyendo los pasos a seguir para empezar a utilizarla. No es necesario descargar la librería ya que se puede incluir utilizando la propia interfaz que ofrece Google con la función "load".

La versión utilizada es Google Feed API v1. Se ofrece como código libre bajo la licencia Apache 2.0.

JQuery

JQuery en una librería JavaScript desarrollada para facilitar el desarrollo de código JavaScript. Ofrece diferentes funcionalidades para facilitar la interacción con el código HTML y el árbol DOM (pudiendo modificarlo añadiendo o eliminando elementos en tiempo de ejecución), cambiar los estilos de los elementos, añadir efectos o incluir eventos de usuario. Además es una librería que ofrece funciones AJAX que permiten hacer llamadas asíncronas al servidor, lo que permite cargar pequeñas partes de información sin tener que actualizar la página entera.

La librería se puede descargar desde la página oficial de JQuery bajo licencia GPL [29]**.** Debido a que esta librería se va a usar junto con JQuery UI es necesario descargarse una versión compatible de ambas librerías. La versión que he utilizado es JQuery 1.5.1.

JQuery UI

JQuery User Interface en una extensión de JQuery. Ofrece una serie de widgets y efectos adicionales como botones, barras de menú, barras de progreso, ventanas, acordeón, selectores de fecha y muchos otros, además de permitir crear un tema personalizado y descargarlo para poder utilizarlo directamente en las aplicaciones web a través de CSS. En la página web oficial hay mucha información acerca de cómo utilizar la librería y de las funcionalidades que ofrece [37].

Al descargar esta librería se descarga a su vez una versión de librería de JQuery original compatible con esta, evitando de esta manera posibles problemas de compatibilidad. La versión que he utilizado para desarrollar es JQuery UI 1.8.12 que se puede descargar de forma gratuita bajo licencia GPL.

Google Maps API

La librería de Google Maps, es la librería de Google que ofrece soluciones para incluir mapas en un sitio web, además de otras opciones como crear aplicaciones basadas en ubicación, personalización de mapas, uso de Street View etc. Se proporciona de dos

formas, una gratuita y otra para negocios en la que se ofrecen funciones mejoradas y una asistencia más completa.

De nuevo, no es necesario descargar la librería ya que se puede cargar mediante la interfaz de Google con la función "load".

JDBC

JDBC (Java DataBase Connectivity) es una librería Java que permite, como su propio nombre indica, realizar operaciones sobre bases de datos mediante el lenguaje Java. Entre las operaciones más generales se encuentran establecer la conexión a la base de datos, enviar sentencias SQL y procesar los resultados. Para utilizar esta API únicamente es necesario importarla dentro del código donde se vaya a utilizar, no es necesario descargarla ya que viene incluida en el JDK. Lo que sí que hay que tener en cuenta es que hace uso del driver que proporciona MySQL que comenté en su apartado correspondiente por lo tanto hay que incluirlo dentro del servidor de aplicaciones.

3.3.6 Software adicional

Para la realización de la memoria del proyecto he utilizado una serie de editores de texto y diagramas.

- Microsoft Office Word 2007 para la toda parte de texto de la memoria.
- StarUML v2.1.1 para la creación de los diagramas UML incluidos en el documento. Desde su página web [38] se puede descargar gratuitamente bajo licencia GPL.
- Microsoft Power Point 2007 para realizar algunos esquemas y diagramas adicionales, así como la presentación.

3.4 Instalación y configuración del entorno de desarrollo

Debido a que no es una tarea sencilla y me resulto difícil dejar en entorno de desarrollo configurado para desarrollar y ejecutar la aplicación, he decidido incluir este apartado para posibles referencias en el futuro, en caso de querer o tener que volver a instalar y configurar los elementos anteriormente comentados. El proceso me llevo bastante tiempo ya que obtuve varios errores sobre todo de compatibilidad entre los diferentes componentes.

Para la realización de esta guía me he basado en la información obtenida de un blog de internet [39] pero con instrucciones propias que he aprendido de mi experiencia personal al instalarlo.

Se empezará listando los recursos necesarios, se continuará comentando como se instalan y por último se explicará cómo se despliega y configura el portal Liferay.

Recursos necesarios

Para comenzar la instalación tendremos que disponer de los siguientes recursos.

- JDK 6 para poder desarrollar en Java.
- Netbeans 6.9.1 como entorno de desarrollo integrado.
- GlassFish 3.0.1 como servidor de aplicaciones.
- El archivo WAR de Liferay 6.0.5.
- liferay-portal-dependencies-5.2.3: Son unos archivos adicionales para preconfigurar el contenedor Web. Se pueden descargar de la siguiente dirección:

  http://downloads.sourceforge.net/lportal/liferay-portal-dependencies-5.2.3.zip

- MySQL 5.6.21 como gestor de base de datos.
- Driver JDBC para Java.
- Xerces-J: Es un analizador de Java que sirve para evitar un problema que existe con el contenedor web por el cual no es capaz de interpretar los archivos de despliegue de algunos portlets. Se puede obtener de la siguiente dirección:

  http://archive.apache.org/dist/xml/xerces-j/Xerces-J-bin.2.9.0.zip

- Portal Pack v3.0.4 : Es el plugin para NetBeans que nos proporcionara una interfaz para crear y desplegar nuevos portlets.

Instalación de los componentes

Una vez listados los recursos, se va a explicar cómo instalarlos. Como trabajo bajo el sistema operativo Windows se explicará para este sistema.

1. En primer lugar hay que instalar el JDK. Se hará en la carpeta por defecto.

2. Una vez instalado el JDK, se procederá a instalar el IDE NetBeans y el servidor de aplicaciones GlassFish ya que se descargan conjuntamente. La instalación se puede realizar en la carpeta que se desee. Se tendrán que indicar ambas rutas durante el proceso de instalación.

3. En tercer lugar se instalará MySQL. La instalación en Windows en muy sencilla, simplemente hay que ejecutar el instalador y seguir los pasos para instalarlo y configurarlo correctamente. Durante la instalación se dará la opción por defecto de instalar además el workbench, por lo que se descarga, instala y configura automáticamente.

4. A continuación se descomprime el zip de dependencias que se ha descargado y se pegan los archivos JAR obtenidos dentro del directorio de instalación de GlassFish en la siguiente ruta: `%GLASSFISH%\glassfish\domains\domain1\lib`.

   También se incluyen en este directorio el driver JDBC Java de MySQL y los archivos xercesImpl.jar y xml-apis.jar de Xerces que se obtienen al descomprimir el archivo zip.

5. Tras pegar todos los archivos en el directorio, se arranca GlassFish con el siguiente comando `%GLASSFISH_HOME%\bin\asadmin start-domain`.

6. Antes de desplegar el WAR de Liferay hay que lanzar el actualizador del servidor GlassFish. Para ello, una vez arrancado el servidor se accederá a la siguiente dirección:

   ```
   http://localhost:4848
   ```

   Se abre la consola de configuración de GlassFish y en el apartado "Update tool", en "add ons" se instalará la aplicación "updatetool". Una vez terminado, se reinicia el servidor y aparece el nuevo ejecutable de update tool en la carpeta `%GLASSFISH_HOME%\bin`.

7. Se lanza el nuevo ejecutable "update tool" que se ha generado, se abre el apartado updates y se espera a que carguen las actualizaciones. Este proceso puede tardar varios minutos. Una vez cargadas se actualiza todo.

8. Tras actualizar las herramientas de NetBeans es necesario hacer la actualización del start-domain con el comando `Asadmin> start-domain --upgrade`

   Este comando te lo pide el propio servidor al tratar de arrancarlo. Es para actualizar el arranque después de haber actualizado el GlassFish. Tras realizar esto ya podemos empezar a desplegar Liferay.

Despliegue de Liferay Portal

9. Con GlassFish iniciado, lo primero que hay que hacer es modificar las propiedades de la maquina virtual de Java donde se ejecutará Liferay. Para ello, en la consola de GlassFish se selecciona del menú de la izquierda: `Configuration > JVM Settings`. Se cambian los valores de MaxPermSize y Xmx a los siguientes:

   ```
   -XX:MaxPermSize=256m
   -Xmx=1024m
   ```

   Se guardan los cambios haciendo clic en guardar y se reinicia GlassFish.

   ```
   %GLASSFISH_HOME%\bin\asadmin stop-domain
   ```

```
%GLASSFISH_HOME%\bin\asadmin start-domain
```

10. Lo siguiente será preparar la base de datos MySQL. Resumiendo, se necesita crear una base de datos, configurar un Pool de Conexiones, y ajustar el .WAR para que utilice este Pool.

Creación de la base de datos

11. Desde una ventana del símbolo del sistema, ejecutar el comando

```
mysql -u root -p
```

Con esto se ejecutará el cliente de MySQL con el usuario root y pedirá la contraseña también root. Se la proporcionamos y se mostrará el prompt mysql>.

12. Para crear la base de datos lportal se escribe el siguiente comando:

```
create database lportal;
```

13. A continuación se creara y se le proporcionarán permisos al usuario lportal sobre la base de datos lportal creada. Este comando proporciona los permisos al mismo tiempo que crea el usuario.

```
grant all on lportal.* to lportal@localhost identified by "lportal";
```

Configuración del pool de conexiones

En este punto se procederá a crear y configurar el pool de conexiones.

14. Entramos a la consola de GlassFish (http://localhost:4848) y se selecciona del árbol de la izquierda `Resources > JDBC > Connections Pool`.

15. Se hace clic en el botón "New…" para crear un nuevo Pool de Conexiones y se rellenan los siguientes campos:

    - Name: LiferayPool
    - Resource Type: javax.sql.ConnectionPoolDataSource
    - Database vendor: MySQL

16. Se hace clic en "Next", de esta forma se mostrarán todas las propiedades de la conexión a la base de datos.

17. En primer lugar se activa el primer check llamado Ping: Enabled. Esto nos permitirá verificar que la conexión se realizó correctamente después de crear el pool de conexiones.

18. A continuación tenemos que buscar las siguientes propiedades en la lista y configurarlas como sigue:

    - URL: jdbc:mysql://localhost/lportal
    - User: lportal
    - Password: lportal
    - UseUnicode: true
    - CharacterEncoding: UTF-8
    - EmulateLocators: true

19. Al guardar la configuración, si todo esta correcto se creará el Pool.

20. Con esto únicamente hemos creado el Pool de conexión, Lo siguiente es registrarlo en el JNDI del servidor. Para ello seleccionamos la opción `Resources > JDBC > JDBC Resources` del menú de la izquierda.

21. Se hace clic en "New…" y se configuran los campos indicados como sigue:

    - JNDI Name: jdbc/LiferayPool
    - Pool Name: LiferayPool

    El nombre debe coincidir con el del Pool de conexiones que se creó en el punto anterior. Se hace clic en "Ok" y pasamos al despliegue de Liferay.

    Despliegue de Liferay

22. Lo primero que hay que realizar es indicarle al archivo WAR de Liferay que se conecte al Pool de conexiones que se ha creado. Para ello hay que añadir una nueva propiedad al archivo `portal-ext.properties` que se puede encontrar dentro de WAR en el directorio `WEB-INF/clases`. Dentro de dicho archivo se añade una nueva línea:

    ```
    jdbc.default.jndi.name=jdbc/LiferayPool
    ```

    Para poder acceder a dicho archivo se cambiará la extensión ".war" por ".rar" y de esta forma podremos acceder al archivo como si de un paquete comprimido se tratase. Al finalizar cambiamos de nuevo las extensiones.

23. Una vez añadida esta propiedad, accedemos de nuevo a la consola de GlassFish. Se accederá a la opción "Applications" del menú izquierdo y se hará clic en "Deploy…" abriéndose una nueva pantalla. Se tendrá que indicar la localización del archivo WAR de Liferay, para ello en la opción "Location" se introducirá la ruta con la ayuda del explorador de Windows. Tras hacer esto, se mostrarán más opciones de configuración de despliegue de Liferay.

24. En la casilla llamada "Context Root:" aparecerá el nombre del archivo .war. Habrá que cambiar este nombre por el signo "/". Tras esto clic en "Ok"

    A continuación se comienza a desplegar Liferay. Este proceso lleva un tiempo por lo que habrá que esperar a que termine antes de ejecutar el portal. Puede que GlassFish nos envíen un mensaje de que se ha terminado de desplegar pero habrá que asegurarse de que haya configurado todos los archivos, bibliotecas, creado las tablas de la base de datos, etc. Se puede ver cómo va el proceso en el log: `%GLASSFISH_HOME%\glassfish\domains\domain1\logs\server.log`.

    Ejecutar Liferay

25. Una vez que se ha completado el despliegue, ya será posible ejecutar Liferay para ello se abrirá la siguiente dirección en el navegador web:

    `http://localhost:8080`

    La carga del portal por primera vez tras lanzar el servidor tarda varios minutos, una vez cargado se puede volver a lanzar sin esperar tanto tiempo.

    Instalación del plugin Portal Pack para NetBeans

    Una vez visto cómo se instala y ejecuta Liferay queda explicar cuál es el proceso para crear y desplegar los portlets dentro del portal.

26. Lo primero que se hará es descomprimir el paquete del plugin Portal Pack para NetBeans.

27. Con NetBeans arrancado, habrá que ir a `Tools > Plugins`. Se esperará a que el gestor de plugins cargue toda la información sobre los plugins disponibles. Se seleccionan todos los que aparecen y se hace clic sobre el botón "Update" para instalarlos.

28. Se vuelve a abrir el gestor de plugins pero esta vez se abre la pestaña "Downloaded" donde se podrán seleccionar los plugins que hayamos descargado en nuestro ordenador. Se hará clic en el botón "Add Plugins" y con el explorador se seleccionará la ruta local donde están localizados los archivos que descomprimimos anteriormente. Para terminar se hará clic en el botón "Install" que instalará el plugin.

    Crear un nuevo portlet

29. Crear un nuevo portlet es muy sencillo una vez que hemos instalado el Portal Pack. Simplemente hay que abrir el menú "New" y seleccionar "Web

Application". Cuando se ofrezca la opción se indicará que es un portlet, pudiendo rellenar algunas opciones como el titulo, nombre, descripción, etc.

30. Una vez terminado el proceso de creación, ya será posible desplegarlo para ver un primer portlet en funcionamiento. Habrá que seleccionar el portlet de la lista de proyectos y hacer clic secundario sobre él, eligiendo la opción "Deploy"

31. Tras unos segundos, se indicará que se ha realizado el despliegue con éxito si todo ha ido correctamente, y entonces se podrá acceder a la pagina del portal para añadirlo. En la barra superior de Liferay se selecciona la opción "Add" y en el menú desplegable se abre "New User Portlet" donde aparecerán los portlets que se creen. Se hace clic en "add" y se añadirá el portlet a la página.

3.5 Conclusión

En este capítulo queda explicado de forma detallada el entorno de desarrollo que se ha utilizado para el desarrollo del proyecto, desde los elementos que lo forman, a su instalación y configuración, por lo que servirá de guía de referencia, en el caso de que se quiera instalar en otro sistema diferente.

En el siguiente capítulo se realizará el análisis del sistema, exponiendo los requisitos que se obtuvieron durante la fase de captura de requisitos, así como los casos de uso.

Capítulo 4

Análisis del sistema

4.1 Introducción

La primera fase del proyecto estuvo destinada al estudio del problema planteado y la búsqueda de posibles soluciones, incluido el estudio y aprendizaje de las diferentes tecnologías disponibles para la resolución del problema y de forma más detallada aquellas que se usan para el desarrollo de este proyecto.

En este capítulo se recoge la información correspondiente a la segunda fase del proyecto, la fase de análisis. En esta fase, una vez que se tiene una idea más centrada de lo que se va a desarrollar y familiarizado con las herramientas y tecnologías a utilizar, se realizará la consecución de casos de uso y requisitos de software.

Este capítulo es un resumen de los documentos de casos de uso y requisitos de software que se obtuvieron en esta fase. Para ver los documentos completos habrá que dirigirse a los apéndices, al final de esta memoria, donde están incluidos dichos documentos de forma completa.

4.2 Descripción general

El producto a desarrollar consiste en un portal de sindicación de información deportiva. Esta información se recogerá mediante diferentes tecnologías que permitirán obtenerla desde diversas fuentes web. El portal estará conformado por distintos portlets. Cada portlet será desarrollado como un proyecto distinto con unos objetivos concretos dentro del portal web. Por lo tanto el proyecto estará formado por un conjunto de proyectos de menor tamaño que una vez unidos y desplegados en el portal ofrecerán la funcionalidad completa del producto a desarrollar.

Su funcionalidad principal será ofrecer información deportiva actualizada sin la necesidad de redactores. Esto es posible debido al uso de las diferentes tecnologías y técnicas estudiadas para el proyecto, como son los servicios web basados en SOAP, los servicios web basados en REST y la tecnología de distribución de contenidos RSS.

Mediante estas tecnologías y técnicas se conseguirá crear un portal de noticias actualizado, generado únicamente mediante la obtención de información ofrecida de forma gratuita por otros portales o sitios web con los mismos intereses que el portal que se va a desarrollar.

4.3 Casos de uso

Los casos de uso proporcionan uno o más escenarios que representan cómo debería interactuar el sistema con los actores o usuarios para conseguir un objetivo. Son una buena técnica para la captura de requisitos potenciales de un nuevo sistema, especialmente indicados para la obtención de requisitos funcionales.

En este apartado se va a hacer un resumen del documento de casos de uso completo que se ha obtenido y que se puede encontrar en el "Apéndice A" al final de la memoria. En este caso se listarán los actores que participan, los casos de uso obtenidos y los diagramas que muestran su interacción.

En primer lugar se van a exponer los actores que participarán en los diferentes escenarios que se plantearán en los casos de uso:

- Visitante: Este actor representará a los usuarios que accedan al portal sindicador de información deportiva para informarse y que no tengan un usuario registrado en el portal.
- Usuario miembro: Este actor representará a los usuarios que posean un usuario registrado en el portal y que estén conectados.
- Administrador: El usuario con este rol será el encargado de gestionar el portal, tanto la disposición de las columnas y los portlets como decidir que portlets aparecerán en la página principal.

Los casos de uso que se han obtenido son los siguientes:

- Registrarse: Recoge el escenario donde un nuevo usuario del portal de registra en este para conseguir su usuario y clave.
- Autenticarse: Recoge el escenario en el cual un usuario registrado se autentica con su usuario y contraseña.
- Visualizar tweets: Recoge el escenario donde un usuario del portal podrá ver información obtenida de Twitter.
    - Cambiar página tablón: Recoge el escenario en el cual el usuario podrá cambiar la página del tablón de tweets para ver mensajes más antiguos.
    - Filtrar tweets por usuario: Recoge el escenario donde el usuario puede filtrar los tweets por usuario.
    - Ver tweet con foto: Recoge el escenario en el cual el usuario puede abrir un tweet con foto para ver esta a mayor tamaño junto con el resto de información.
- Visualizar feeds RSS: Recoge el escenario donde un usuario del portal podrá ver información obtenida mediante la tecnología RSS.
    - Hacer foco: Recoge el escenario en el cual un usuario podrá parar el bucle automático del slideshow si desea ver una noticia en particular.
    - Acceder noticia completa: Recoge el escenario donde un usuario podrá leer la noticia completa si le interesa, una vez ha leído el resumen.
    - Seleccionar canal RSS: Recoge el escenario en el cual un usuario podrá cambiar el canal RSS a visualizar por otro de los ofrecidos.
- Visualizar inf. Mundial 2014: Recoge el escenario donde un usuario del portal podrá ver información sobre el mundial de futbol 2014.
    - Visualizar grupos: Recoge el escenario en el cual un usuario podrá informarse sobre el Mundial de Brasil 2014, viendo los grupos con sus correspondientes selecciones.
    - Visualizar plantillas: Recoge el escenario donde un usuario podrá informarse sobre las plantillas con las que participaron las diferentes selecciones en el Mundial.
    - Visualizar lista estadios: Recoge el escenario en el cual un usuario podrá ver una lista de estadios donde se jugaron los partidos del mundial.
    - Visualizar info estadios: Recoge el escenario donde un usuario podrá acceder a la información de un estadio en concreto.
    - Visualizar goleadores: recoge el escenario en el cual un usuario podrá informarse sobre los máximos goleadores del Mundial de Brasil 2014.
- Visualizar inf. Liga LFP: Recoge el escenario donde un usuario podrá informarse sobre la liga de futbol profesional española.
    - Visualizar clasificación: Recoge el escenario en el cual un usuario podrá informarse sobre la clasificación de los equipos de primera división mediante una tabla ordenable con la información.

  - Visualizar estadísticas: Recoge el escenario donde un usuario podrá informarse sobre las estadísticas de la liga LFP en cuanto a goles, asistencias y amonestaciones mediante una tabla ordenable.
  - Visualizar plantillas liga: Recoge el escenario en el cual un usuario podrá informarse sobre las plantillas de jugadores que componen cada equipo de primera división así como su información correspondiente a la competición.
- Gestionar columnas portal: Recoge el escenario donde un usuario administrador podrá cambiar la disposición de las columnas del portal.
- Gestionar posición portlets: Recoge el escenario en el cual un usuario administrador podrá cambiar la posición de un portlet en el portal.
- Desplegar portlet administrador: Recoge el escenario donde un usuario administrador podrá desplegar un nuevo portlet en el portal.
- Replegar portlet administrador: Recoge el escenario en el cual un usuario administrador podrá replegar un portlet desplegado en el portal.
- Crear páginas personales: recoge el escenario donde un usuario miembro podrá crear páginas personales donde será el encargado de gestionar tanto los portlets que desee que aparezcan como la posición de los mismos en la página y la distribución de columnas.
- Desplegar portlet miembro: Recoge el escenario en el cual un usuario miembro podrá añadir un nuevo portlet a una de sus páginas personalizadas.
- Replegar portlet miembro: Recoge el escenario donde un usuario miembro podrá replegar un portlet de una de sus páginas personales.

A continuación se muestran los diagramas de casos de uso.

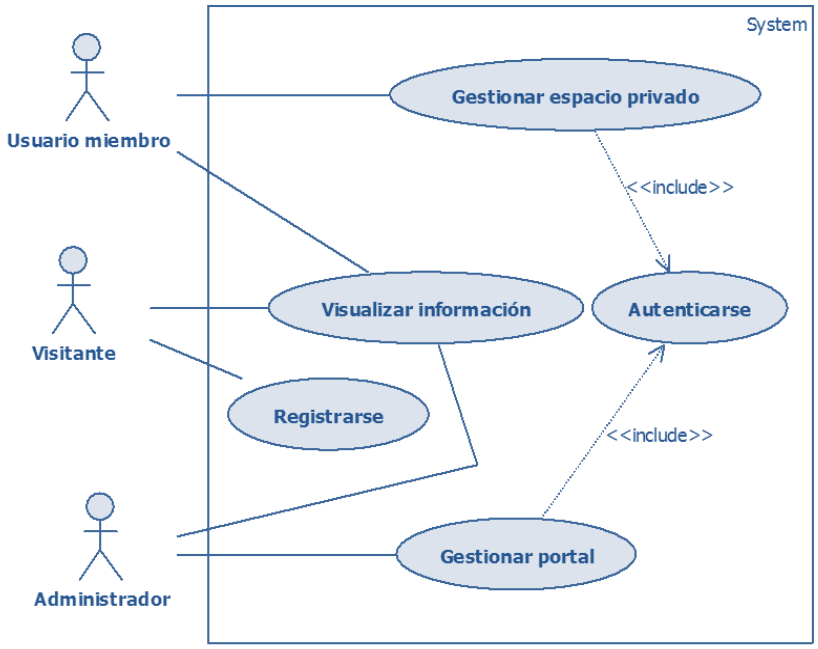

Ilustración 8. Diagrama de casos de uso general.

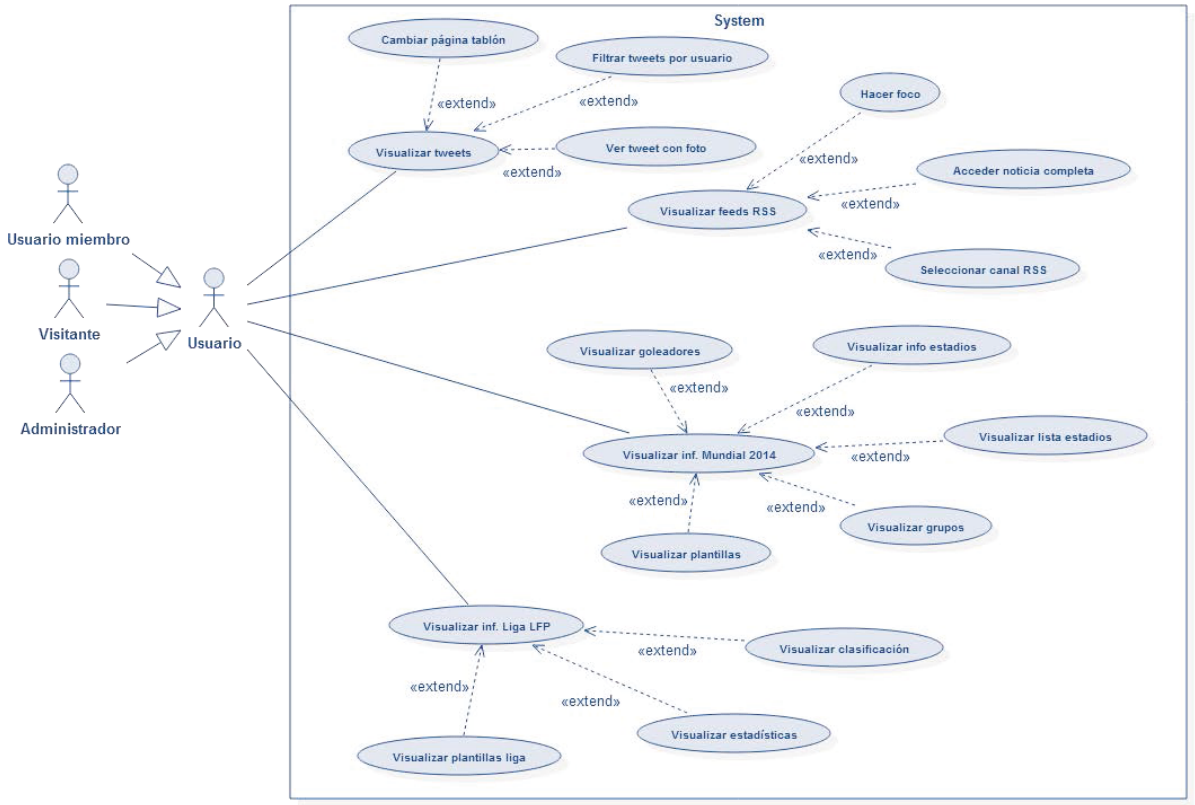

Ilustración 9. Diagrama de casos de uso: visualizar información.

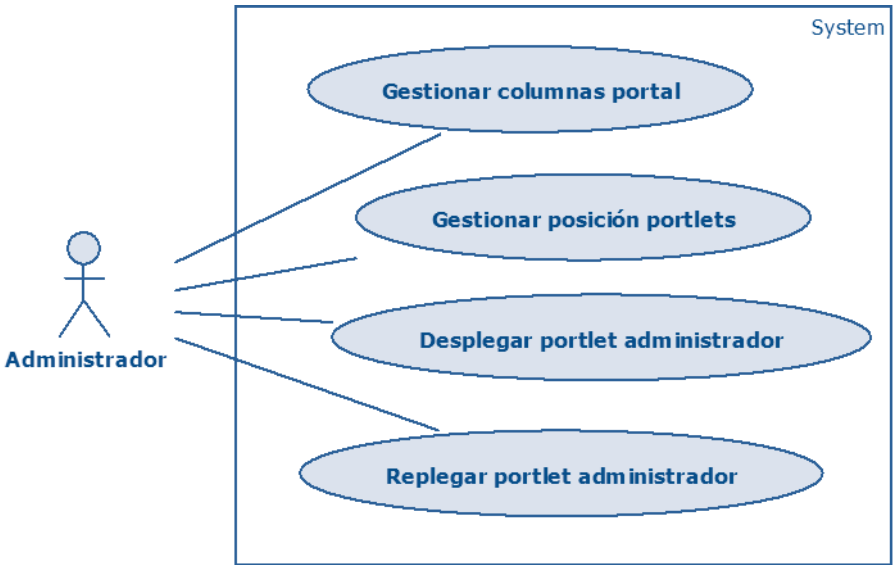

Ilustración 10. Diagrama de casos de uso: gestionar portal.

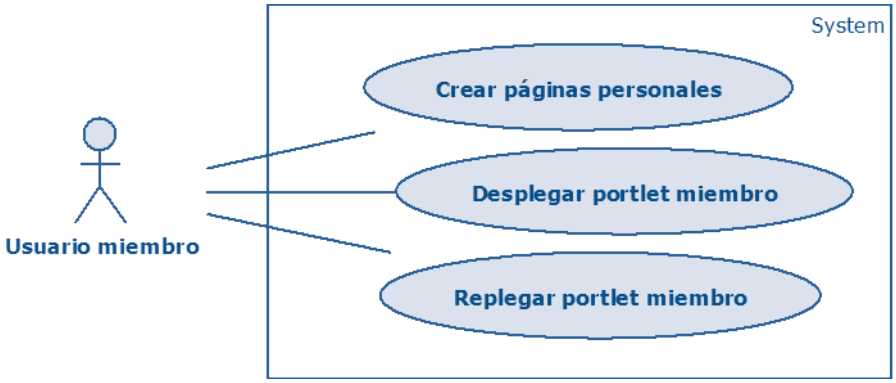

Ilustración 11. Casos de uso: gestionar espacio privado.

4.4 Requisitos de software

Los requisitos de software definirán los diferentes aspectos que debe cumplir la aplicación a desarrollar, sirviendo como guía de referencia al desarrollador. A su vez permitirán comprobar que a la finalización del proyecto se han cumplido los diferentes objetivos que se propusieron en la fase de análisis.

Los requisitos se han dividido en dos grupos, requisitos funcionales y no funcionales. Los requisitos funcionales definen, como su propio nombre indica, las funcionalidades que ofrece la aplicación al usuario. Los no funcionales definen otros aspectos sobre el sistema, usabilidad de la aplicación e interfaz. Precisamente los requisitos no funcionales se han dividido en estos tres grupos, requisitos de sistema, de usabilidad y de interfaz. Los requisitos de usabilidad se han obtenido basándose en el libro The Design of Sites: Patterns for Creating Winning Sites [40]. En ellos aparece en cursiva junto a la descripción, el identificador y el nombre del patrón en el que está basado.

A continuación se exponen los requisitos con su identificador, nombre y descripción. En el documento de requisitos completo del "Apéndice B", se exponen clasificados por cada portlet para que se vean todos en contexto, pero en este apartado los expondré como una lista ordenada según su identificador, de esta forma no quedará tan cargado y se evitará repetir algunos requisitos que pertenecen a varios portlets.

4.4.1 Requisitos funcionales.

RF-001	**Registrarse en el portal**

Los visitantes del portal podrán registrarse en el portal si lo desean, convirtiéndose en usuarios del mismo.

RF-002 Autenticarse en el portal

Los usuarios del portal previamente registrados podrán acceder al mismo mediante un formulario de autenticación. Esto le permitirá realizar las opciones propias de su rol.

RF-003 Recordar identificador y contraseña

El usuario que se va a autenticar podrá marcar una opción para que se mantengan los datos de su autenticación (identificador y contraseña). De esta manera, la próxima vez que el mismo usuario vaya a autenticarse en el portal no será necesario reintroducirlos.

RF-004 Desconexión del portal

Un usuario que se encuentre autenticado en el portal podrá pulsar un enlace que le desconectará del portal.

RF-005 Gestionar espacio personal

El usuario podrá personalizar su espacio personal mediante la barra de administración que aparecerá en la parte superior del portal una vez autenticado. Podrá crear nuevas páginas personales, añadir y eliminar nuevos portlets en sus propias páginas y colocarlos en la disposición que desee.

RF-006 Modificar disposición página principal

El usuario con derechos de administración podrá modificar la disposición de los portlets de la página principal mediante la barra de administración que aparecerá en la parte superior del portal una vez autenticado. Podrá desplegar nuevos portlets y replegar los ya desplegados. Todos los usuarios del portal verán esta disposición cuando accedan al portal a través del navegador.

RF-007 Modificar el esquema de columnas del portal

El usuario con derechos de administración podrá modificar el número y la disposición de las columnas del portal mediante la barra de administración que aparecerá en la parte superior del portal una vez autenticado. Todos los usuarios del portal verán esta disposición de columnas cuando accedan al portal a través del navegador.

RF-008 Visualizar información vía servicio REST

El usuario podrá visualizar los mensajes escritos en Twitter por los equipos de fútbol de primera división y principales medios deportivos, obteniendo la información de un servicio web basado en REST.

RF-009 Visualizar tweets en general

El usuario podrá ver los tweets de todos los usuarios de Twitter que estén siendo seguidos por la cuenta de Twitter de seguimiento. Esta cuenta consistirá en una cuenta normal de Twitter que seguirá a los usuarios de los equipos de la liga de fútbol de primera división y de los principales medios de comunicación deportivos. Mediante las APIs de Twitter definidas en el requisito RNF-RS-010, se obtendrá la información que se desea mostrar.

RF-010 Filtrar tweets por usuario

El usuario podrá visualizar los tweets filtrados por usuario para facilitar la visualización de los mensajes de un usuario concreto.

RF-011 Visualizar tweets antiguos

Los tweets mostrados se dividirán en páginas de diez mensajes. Los tweets más nuevos serán los mostrados en la página principal (la página 1). El usuario podrá seleccionar la página que desee visualizar para ver los tweets más antiguos.

RF-012 Visualizar tweet con foto

Los tweets que dispongan de fotografía podrán ser visualizados individualmente de forma que la imagen se muestre a mayor tamaño junto con el resto de información.

RF-013 Visualizar noticias vía RSS

El usuario podrá visualizar noticias deportivas obtenidas a través de los canales RSS de los principales medios de información.

RF-014 Seleccionar feed RSS a mostrar

El usuario podrá seleccionar el canal RSS del medio de información que desee visualizar. Una vez seleccionado, el portlet cargará las últimas noticias introducidas en ese canal y las mostrará.

RF-015 Seleccionar noticia a mostrar

El usuario, mediante el ratón, podrá seleccionar la noticia sobre la que se hará foco de la lista de noticias mostradas. Las noticias se mostrarán en forma de slideshow, que irá cambiando la noticia mostrada cada cierto tiempo. Cuando se realice el foco sobre una noticia el giro automático se detendrá hasta que se retire el ratón.

RF-016 Acceder a la noticia completa

El usuario podrá acceder a la noticia completa mediante un hipervínculo en su titular, siendo redireccionado a la página de la noticia en el sitio web de la fuente.

RF-017 Visualizar información vía servicio web basados en SOAP

El usuario podrá visualizar información sobre el Mundial de Brasil 2014 y la liga de fútbol profesional, obtenida a través de un servicio web basado en SOAP. A partir de un fichero WSDL se obtendrá toda la información.

RF-018 Visualizar plantilla selecciones Mundial 2014

El usuario podrá visualizar las plantillas de cada selección del Mundial de fútbol 2014 divididas por posiciones de los jugadores. Se accederá haciendo clic sobre cualquiera de los equipos de la pagina de grupos.

RF-019 Visualizar estadios Mundial 2014

El usuario podrá visualizar la información sobre los estadios del Mundial 2014. Podrá seleccionar el estadio sobre el que desee informarse a partir de la tabla de estadios.

RF-020	Visualizar grupos Mundial 2014

El usuario podrá visualizar la información sobre los grupos del Mundial 2014.

RF-021	Volver a página principal

El usuario podrá volver a la página principal del portlet (página de grupos) mediante un botón.

RF-022	Visualizar goleadores Mundial 2014

El usuario podrá visualizar los máximos goleadores del Mundial 2014

RF-023	Visualizar clasificación primera división

El usuario podrá visualizar la información correspondiente a la clasificación de los equipos de la liga LFP de primera división.

RF-024	Ordenación de tabla

El usuario podrá ordenar la información de las tablas de información de la liga LFP, por cada una de las columnas de esta para ver la información ordenada descendente o ascendentemente.

RF-025	Visualizar clasificación de goleadores

El usuario podrá ver la información de goleadores de la liga LFP.

RF-026	Visualizar clasificación de asistencias

El usuario podrá ver la información de asistentes de la liga LFP.

RF-027	Visualizar clasificación de amonestaciones

El usuario podrá ver la información de amonestaciones de la liga LFP.

RF-028	Visualizar equipos

El usuario podrá ver una tabla con los equipos de primera división.

RF-029	Visualizar plantilla de equipo

El usuario podrá ver la plantilla completa de un equipo seleccionado.

RF-030	Volver a página de equipos

El usuario podrá ver volver a la página de equipos desde cualquier plantilla.

4.4.2 Requisitos no funcionales.

4.4.2.1 Requisitos del sistema.

RNF-RS-001	Compatibilidad con navegadores web

La aplicación deberá ser compatible con Internet Explorer 7.0 y superior y Mozilla Firefox 7.0 y superior.

RNF-RS-002 | Bases de datos

La aplicación utilizará MySQL 5.6 como sistema gestor de bases de datos.

RNF-RS-003 | Portal gestor de contenidos

El portal gestor de los diferentes portlets será Liferay portal 6.0.5.

RNF-RS-004 | Servidor Web

El servidor web sobre el que se ejecutará la aplicación será GlassFish 3.0.1

RNF-RS-005 | Especificación portlets

La versión a utilizar de la especificación de los portlets Java será la JSR 286: Portlet Specification 2.0.

RNF-RS-006 | IDE

El desarrollo de la aplicación se realizará con la ayuda del IDE NetBeans 6.9.1.

RNF-RS-007 | Lenguaje de desarrollo del lado del servidor

El código del lado del servidor será desarrollado en Java (JSP2.2. , Servlet3.0) Se utilizará el jdk 1.6.0 que incluye el jre6 para la ejecución del código java.

RNF-RS-008 | Librerías de Interfaz

Para desarrollar la interfaz de usuario, se utilizarán la librerías JQuery v1.5.1 y JQuery UI v1.8.12.

RNF-RS-009 | Uso tecnología AJAX

Para la elección de usuario y página a mostrar en el portlet de Twitter y para la actualización automática del tablón de tweets, se utilizará la función POST de JQuery para realizar llamadas asíncronas al servidor y evitar la actualización completa del portal.

RNF-RS-010 | Uso de las APIs REST y Stream de Twitter

Para obtener la información de Twitter se usarán las APIs ofrecidas por el mismo. La API REST de Twitter permite obtener la información mediante una simple interfaz. El API Stream de Twitter permite obtener información en tiempo real.

RNF-RS-011 | Uso de la librería Twitter4j

Se utilizará la versión 3.0.3 de la librería Twitter4j para manejar las APIs REST y Stream de Twitter en la aplicación.

RNF-RS-012 | Almacenamiento de tweets en base de datos

El sistema almacenará los mensajes en una base de datos para poder visualizarlos por los diferentes criterios (General o por usuario y antigüedad).

RNF-RS-013	Actualización automática portlet REST

El sistema actualizará el portlet que visualiza tweets para mantenerlo actualizado en todo momento (cada 15 segundos).

RNF-RS-014	Uso del API Google Feed

Para la lectura de los feeds RSS a partir de su URL, se utilizará el API Google Feed v1.

RNF-RS-015	Actualización automática portlet RSS

El sistema actualizará automáticamente las noticias mostradas en el portlet RSS a los más recientes cada 15 minutos.

RNF-RS-016	Uso de JQuery

Para el desarrollo de JavaScript y la creación de la interfaz se utilizará la librería JQuery v1.5.1 y JQuery UI v1.8.12.

RNF-RS-017	Uso EJB

Como varios portlets van a acceder al mismo servicio web, se utilizara un EJB como cliente del servicio, de esta forma se evitará tener que crear un cliente por cada portlet, así como la repetición de las clases generadas por el cliente en cada uno de ellos.

4.4.2.2 Requisitos del usabilidad.

RNF-RU-001	Abrir los Hipervínculos en pestañas diferentes

Los hipervínculos a páginas externas se abrirán en pestañas diferentes para que los usuarios no pierdan el contexto del portal. (Patrón K8 External links).

RNF-RU-002	Hipervínculos fáciles de identificar

Todos los hipervínculos deben ser fáciles de identificar. Deberán diferenciarse de forma clara cuando un hipervínculo ha sido visitado y cuando no cambiando el color. Los hipervínculos estarán subrayados y también cambiarán de color cuando se pase el ratón por encima. (Patrón K10 Obvious links).

RNF-RU-003	Ajuste de maquetación automática

La maquetación del lector de tweets cambiará en tiempo de ejecución para ajustarse y optimizar su visualización, teniendo en cuenta el tamaño de la columna donde se localice.

RNF-RU-004	Estilo de tweets

La información de los tweets que se muestren tendrá un formato y color similar al que usa Twitter para que sea fácil diferenciar cada parte del mensaje, haciendo estos reconocibles para los usuarios del programa.

RNF-RU-005 | Ventana modal para Tweets con foto

Los tweets que dispongan de imagen se podrán expandir para ver esta a un tamaño mayor junto con el resto de información. Para hacer esto se hará uso de una ventana modal que permitirá al usuario volver al contexto anterior simplemente cerrando esta ventana. (Patrón H6 Helping users complete tasks).

RNF-RU-006 | Forma corta de las noticias RSS

En el portlet de noticias obtenidas mediante RSS se mostrará el titular junto una imagen representativa si es ofrecida por el feed y un resumen de la noticia. En el supuesto que el usuario desee leer la noticia completa, se accederá al sitio web de la fuente mediante un hipervínculo. (Patrón A2 News mosaics).

RNF-RU-007 | Usar nombres descriptivos en los hipervínculos

Los nombres utilizados en los hipervínculos serán los titulares de la noticias. De esta manera serán intuitivos. (Patrón K9 Descriptive, Longer Link Names).

RNF-RU-008 | Slideshow para los Feeds

El sistema mostrará las noticias más recientes de los feeds mediante un slideshow que cambiará la noticia a mostrar de forma automática cada 7 segundos en un bucle circular.

RNF-RU-009 | Ajuste de la visualización Feeds

El sistema analizará el tamaño de la columna donde se encuentre el portlet RSS y adecuará su disposición a este tamaño para facilitar su visualización.

RNF-RU-010 | Camino de navegación

Se mostrará el camino recorrido durante la navegación en el portlet de información del mundial 2014 para saber en qué lugar se encuentra el usuario en cada momento. (Patrón K6 Bread Crumbs).

4.4.2.3 Requisitos de interfaz.

RNF-RI-001 | Formulario de autenticación

El portlet de autenticación estará formado por un campo de correo electrónico que funcionará como identificador y un campo contraseña. Además dispondrá de un chekbox para seleccionar si recordar los datos del autenticación. También aparecerá un enlace que conducirá al formulario de registro y otro para recuperar contraseña.

RNF-RI-002 | Formulario de registro

El formulario de registro estará compuesto por los campos nombre, segundo nombre, apellido, nombre de usuario, dirección de correo(ID), fecha de nacimiento, género y texto de verificación.

RNF-RI-003 | Disposición portlet de Twitter vía REST

El portlet de mensajes de Twitter tendrá un tablón central donde se mostrarán los tweets. En la parte superior del tablón habrá un menú desplegable con el que se seleccionará el usuario del cual se desea ver los mensajes. En la parte inferior del tablón habrá otro menú desplegable con el que se seleccionará la página de tweets a mostrar, siendo la número uno la de los tweets más modernos.

RNF-RI-004 | Información de Tweet

De cada tweet se mostrará la imagen de usuario, el nombre de usuario, el texto y la fecha.

RNF-RI-005 | Disposición portlet de noticias vía RSS

El portlet de noticias que muestre feeds RSS se compondrá de dos zonas: La parte derecha contendrá los titulares de las últimas noticias. La parte izquierda tendrá tres subapartados, el titular de la noticia, la fotografía y un pequeño resumen. Cuando se pase el ratón por encima de un titular de la zona derecha aparecerá su titular, fotografía y resumen correspondiente en la parte izquierda. Además dispondrá de un menú desplegable para seleccionar el feed RSS del medio informativo que se desee entre los ofrecidos.

RNF-RI-006 | Acceso a las plantillas de las selecciones del Mundial 2014

Para el acceso a las plantillas de cada selección nacional, el portlet dispondrá los grupos oficiales del Mundial 2014 conteniendo las selecciones correspondientes en cada uno. El nombre de cada selección será un hipervínculo que conducirá hasta su plantilla.

RNF-RI-007 | Plantillas selecciones nacionales

Para visualizar las plantillas completas de las selecciones nacionales se dispondrá en un tablón a los jugadores divididos por su posición en el campo. Para regresar a la pantalla de elegir selección nacional existirá un enlace "volver" o se podrá regresar también mediante el camino de navegación.

RNF-RI-008 | Acceso a los estadios del Mundial 2014

Para el acceso a la información de cada estadio, el portlet dispondrá de un listado de los estadios con su foto. Cada foto será un hipervínculo a toda la información sobre ese estadio.

RNF-RI-009 | Información de los estadios del Mundial 2014

La información de los estadios estará compuesta por el nombre del estadio, la ciudad, la capacidad del estadio, el enlace a la página web sobre el estadio en wikipedia, la imagen del estadio y el mapa de Google Maps con la situación del estadio. Para regresar a la pantalla de elección de estadio existirá un enlace "volver" o se podrá regresar también mediante el camino de navegación.

RNF-RI-010 | Información de los goleadores del Mundial 2014

La Información de los goleadores constará de un listado de los diez máximos goleadores del Mundial 2014. De cada goleador se mostrará el nombre, la bandera de su nacionalidad y el número de goles conseguidos.

RNF-RI-011 | Tabla ordenable

Se deberá implementar una tabla ordenable que permita ordenar su contenido ascendente y descendentemente dependiendo de cada columna.

RNF-RI-012 | Información tabla clasificación LFP

La tabla de clasificación de la liga LFP deberá mostrar la posición de cada equipo con un número, una imagen que señale si el equipo está en Europa o en puestos de descenso, el nombre del club, una imagen del escudo, los partidos jugados, los partidos ganados, los partidos empatados, los partidos perdidos, los goles a favor, los goles en contra y la diferencia de goles de cada equipo.

RNF-RI-013 | Disposición portlet estadísticas LFP

El portlet de estadísticas de la LFP se compondrá de un tablón central donde se mostrará la tabla correspondiente con la información. En la parte superior tendrá tres botones para poder cambiar la tabla que se quiera visualizar en el tablón, ya sea la clasificación de goleadores, la de asistencias o la de amonestaciones.

RNF-RI-014 | Información tabla goleadores

La tabla de clasificación de goleadores deberá mostrar la foto, el nombre, el escudo del equipo al que corresponde y el número de goles de cada jugador.

RNF-RI-015 | Información tabla asistencias

La tabla de clasificación de asistencias deberá mostrar la foto, el nombre, el escudo del equipo al que corresponde y el número de asistencias de cada jugador.

RNF-RI-016 | Información tabla amonestaciones

La tabla de clasificación de amonestaciones deberá mostrar la foto, el nombre, el escudo del equipo al que corresponde, el número de tarjetas rojas y el número de tarjetas amarillas de cada jugador.

RNF-RI-017 | Disposición portlet plantilla LFP

El portlet de plantillas de los equipos de la LFP se compondrá de una pantalla inicial con los escudos de los equipos de primera división, sirviendo estos como links que permitirán acceder a la información de las plantillas de cada equipo. Esta información será presentada en una tabla ordenable. Desde cualquier pantalla de plantilla se podrá volver a la página principal mediante un botón de volver.

RNF-RI-018	Información tabla plantilla

La tabla de la plantilla de un equipo deberá mostrar el dorsal, la foto, el nombre, la posición, el número de goles, el número de asistencias, el número de tarjetas rojas y el número de tarjetas amarillas de cada jugador.

4.5 Conclusión

Con los casos de uso y los requisitos de software obtenidos y expuestos en este capítulo quedan definidos de forma clara y concisa los objetivos que deberán cumplirse tras finalizar la fase de desarrollo.

En el siguiente capítulo se va a exponer la documentación obtenida en la fase de diseño, incluyendo la arquitectura del sistema y que será la fase previa que dejará todo preparado antes de pasar a la fase de desarrollo.

Capítulo 5

Diseño del sistema

5.1 Introducción

Una vez terminada la fase de análisis donde se obtuvieron los casos de uso del sistema a desarrollar y los requisitos que el proyecto debe cumplir, se va a comenzar con la fase de diseño del sistema. En esta fase se va a explicar la arquitectura física y la arquitectura lógica del sistema. Debido al elevado número de componentes de los que el sistema estará formado, cuando se explique la arquitectura lógica, se comenzará con una descripción general y posteriormente se irá descomponiendo en los diferentes módulos que forman parte de ella, describiendo cada uno de una forma más detallada.

De cada módulo, a parte de su arquitectura lógica de una forma más detallada, se realizará también su diseño estático y dinámico, así como el diseño de la base de datos utilizada por los componentes a desarrollar.

5.2 Arquitectura Física

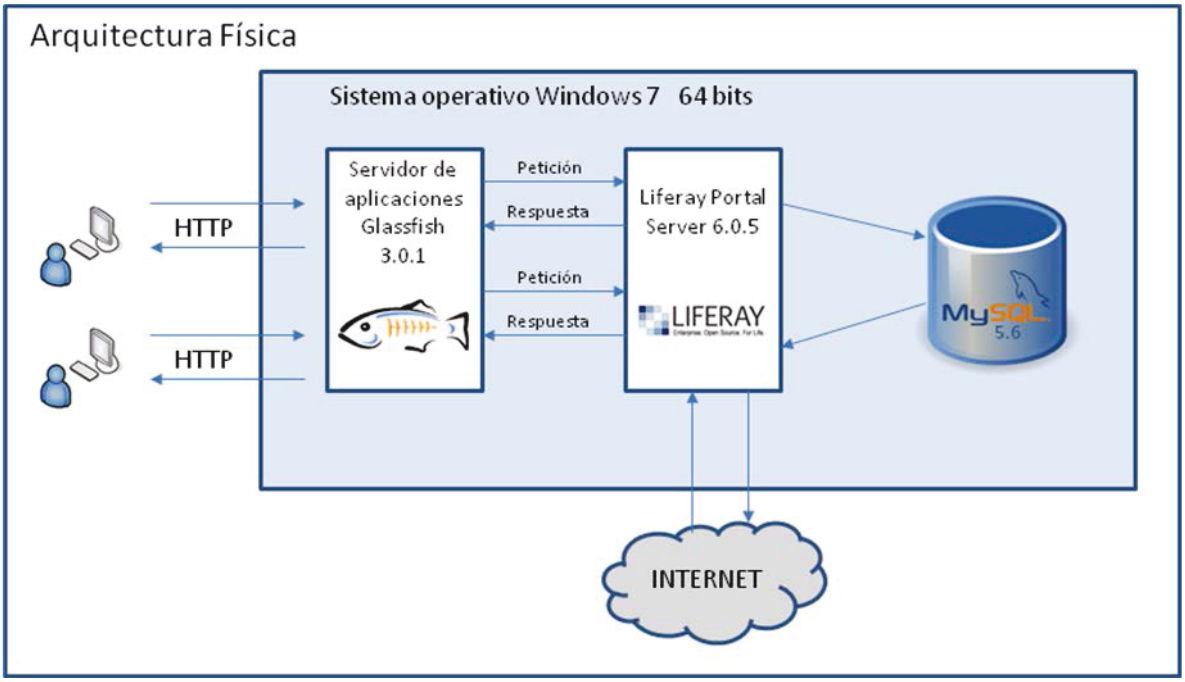

Ilustración 12 Arquitectura física.

Los diferentes módulos y componentes que conforman el sistema se ejecutarán sobre un servidor de aplicaciones GlassFish que estará instalado en un sistema operativo Windows. Como el proyecto se basa en un portal de sindicación de información formado por componentes Portlet de Java, es necesario tener instalado un servidor del portal que soporte este tipo de componentes. En el caso a tratar se va a utilizar Liferay Portal. Realmente el módulo Liferay es una aplicación que es desplegada sobre el servidor de aplicaciones elegido, pero una vez cargada sobre él se puede tratar como si realmente fuera un servidor.

El sistema necesita de una base de datos donde el portal pueda generar el esquema necesario para el mantenimiento del portal, tareas tales como la gestión de usuarios, roles, páginas del portal y la gestión de los portlets disponibles para el despliegue entre otras. Para ello se ha instalado un servidor de bases de datos MySQL. Además del propio portal, también harán uso de esta base de datos algunos de los diferentes módulos y portlets a desarrollar.

Debido a que el Proyecto se basa en obtener información para disponerla en las páginas del portal, es necesario que el sistema tenga acceso a Internet para verlo a su máximo rendimiento.

El sistema estará montado en un entorno local, instalado sobre mi propio ordenador personal. De este modo el sistema no dará acceso al público en general, pero sí que será accesible dentro de un entorno de red local en el que el ordenador esté conectado, conociendo la dirección ip local del equipo.

Las versiones utilizadas de los elementos que conforman el sistema son las siguientes:

- Como servidor de aplicaciones se utilizará GlassFish 3.0.1.
- Como portal contenedor de Portlets se utilizará Liferay Portal 6.0.5.
- Como se van a desarrollar los componentes Portlet de Java será necesario instalar un kit de desarrollo de Java, en este caso se utilizará el JDK 1.6.0.
- Para la base de datos se instalará la versión 5.6 de MySQL.

En cuanto a librerías a utilizar en el desarrollo de los componentes del sistema se utilizarán:

- JDBC, una librería que permite realizar operaciones sobre bases de datos mediante el lenguaje Java. Entre las principales operaciones se encuentran establecer la conexión a la base de datos, enviar sentencias SQL y procesar los resultados.
- JNDI, es un servicio de nombres que permite a un componente localizar otros componentes o recursos. Concretamente se utilizará para localizar los elementos EJB creados durante la realización del proyecto desde los portlets cliente.
- JQuery, una librería que ofrece facilidades en el uso del lenguaje JavaScript, como por ejemplo en el manejo del árbol DOM, añadiendo y eliminando elementos, añadiendo evento. Además ofrece funciones AJAX para realizar llamadas asincrónicas al servidor.
- JQuery User Interface, una extensión de JQuery que ofrece una serie de widgets y efectos para diseñar las interfaces de usuario de los componentes Portlet.
- Twitter4j, una librería Java no oficial que proporciona funciones de más alto nivel para las diferentes operaciones que ofrece el API REST de Twitter.
- Google Feeds API, una librería JavaScript de Google que permite descargar cualquier Atom público o fuente de medios RSS usando únicamente JavaScript.
- Google Maps API, otra librería de Google que ofrece soluciones para incluir mapas en un sitio web. En este caso concreto se utilizará para introducir un mapa de la zona de los estadios del mundial de fútbol de Brasil 2014 en el portlet.

El sistema , al lanzar el servidor de aplicaciones carga automáticamente el módulo de Liferay, que accede a la base de datos para obtener toda la información necesaria para arrancar el portal. Una vez arrancado los usuarios tienen acceso al portal a través de su navegador. Los componentes portlet actualizan periódicamente la información del portal accediendo a Internet.

Una vez descrita la arquitectura física del sistema se comenzará con la descripción de la arquitectura lógica.

5.3 Arquitectura Lógica

En este apartado se realizará una descripción general de la arquitectura lógica del sistema, dejando para los siguientes apartados la descripción de la lógica da cada uno de los módulos mostrados en la visión general. De esta manera quedará definida de una manera mucho más clara y fácil de comprender que amontonando todos los módulos en un mismo esquema.

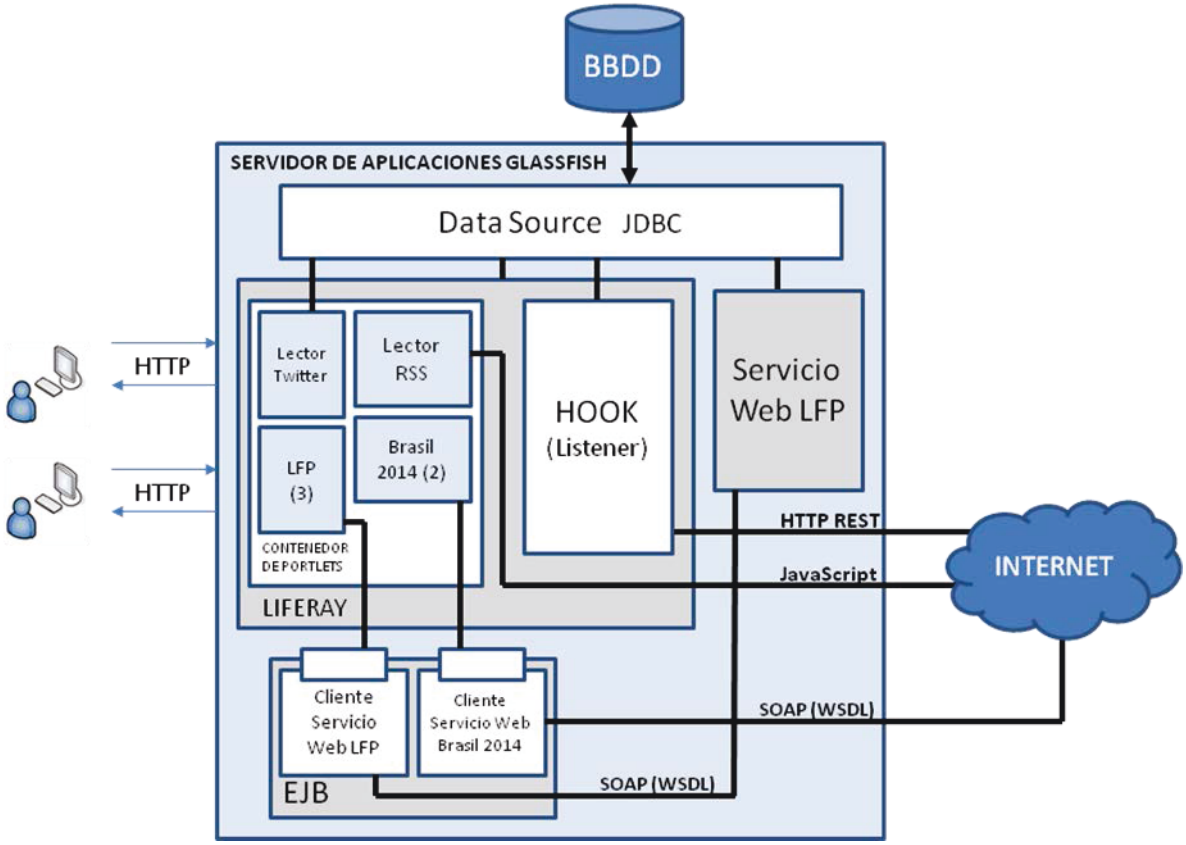

Ilustración 13. Arquitectura lógica

En el esquema puede observarse todos los módulos desarrollados trabajando en conjunto para ofrecer la funcionalidad final del sistema, crear un portal de sindicación de información deportiva. Todos estos módulos se ejecutan sobre el servidor de aplicaciones GlassFish que se comentó en la arquitectura física en el apartado anterior.

Para comenzar a explicar cómo está compuesto el sistema se debe comentar que existen tres módulos principales: el módulo de Liferay, el módulo EJB y un servicio web basado en SOAP.

El módulo de Liferay es el módulo que permite que cuando los clientes accedan al servidor de aplicaciones, estos vean un portal web. Este módulo, una vez desplegado ya funciona como portal aún sin haber creado ningún portlet propio. Un portal Liferay se compone de módulos Portlet para ofrecer la información a los usuarios del mismo pero

también dispone de otros dos tipos de componentes con funcionalidades diferentes: los themes y los hooks.

Los themes (en español temas) son usados para crear apariencias visuales para todos los elementos de un mismo portal, es decir para que se vean todos con la misma apariencia, tanto menús como cabeceras de los portlets. En mi caso no se creará ningún tema para el portal, se definirá el tema en cada portlet a desarrollar mediante la librería JQuery User Interface.

Los hooks son componentes que tienen varias funciones, por un lado permiten modificar las propiedades de un portal de forma no intrusiva es decir sin tocar código que realizaría un cambio definitivo, ya que la principal ventaja de los hooks es que se ponen en funcionamiento tan pronto como se despliegan y dejan de funcionar en cuanto se repliegan, no cambian nada definitivamente. La segunda funcionalidad es la de añadir alguna funcionalidad adicional a un portal. Como puede verse en el esquema, en mi sistema estoy utilizando un hook. En mi caso se va a utilizar para añadir una funcionalidad que se desea que empiece funcionar nada más arranque el servidor: un listener de tweets a través de las APIs REST y Stream de Twitter. Este hook va a quedarse escuchando para recibir los mensajes que escriben los usuarios de Twitter que se desean y los va a introducir en la base de datos para que posteriormente el portlet lector de Twitter los muestre.

Como ya se ha comentado un poco más arriba, los portlets son los componentes que van a ocupar cierta cantidad de pantalla para mostrar la información a los usuarios. Para gestionar que los portlets funcionen correctamente, éstos siempre se encuentren dentro de un contenedor de portlets. El contenedor de portlets se encarga de gestionar todas las peticiones que lleguen para realizar cambios en un portlet. Se encarga de pasar la petición al portlet correspondiente y de recoger el fragmento de código que estos crean para devolverlo en la respuesta. En el esquema, dentro de este contenedor se pueden ver los 7 portlets que se van a desarrollar para crear el sindicador de información. A continuación se comentará brevemente la función de cada uno de ellos.

El portlet lector de RSS va a leer de diferentes canales RSS a través de Internet. Estos canales serán predefinidos durante el desarrollo permitiendo a los usuarios elegir que canal desean leer en cada momento. El canal seleccionado será visualizado mediante un slideshow creado desde cero a partir de la librería Google Feed y de JQuery. Este portlet estará totalmente desarrollado en JavaScript.

El portlet lector de mensajes de Twitter va a leer y mostrar los mensajes de Twitter que previamente el hook explicado arriba ha ido introduciendo en la base de datos. Como el hook esta constantemente escuchando por nuevos tweets, el portlet estará actualizado casi al instante permitiendo a los usuarios además leer mensajes antiguos.

Los otros 5 portlets obtienen la información de dos servicios web basados en SOAP. Los tres portlets que mostrarán información de Liga de fútbol profesional (LFP) accederán a la información de un servicio web local que implementaré y gestionaré como proveedor, mientras que los otros portlets que obtienen la información del mundial de fútbol de Brasil lo harán de un servicio web externo cuyo archivo descriptor WSDL se encuentra en la siguiente dirección. http://footballpool.dataaccess.eu/data/info.wso?wsdl.

En un principio se pensó en crear cada portlet como un cliente del servicio web correspondiente pero finalmente se descartó esa idea ya que de ese modo, cada cliente debería generar todas las clases para el acceso al servicio a partir del descriptor WSDL. Con lo cual, para acceder al mismo servicio se estaría generando una gran cantidad de código repetido que realmente sólo sería necesario generarlo en una ocasión. De esta idea surge a utilización del siguiente módulo principal nombrado al principio de la explicación: el módulo EJB.

El módulo EJB está compuesto por dos EJBs de sesión sin estado que harán la función de clientes de los respectivos servicios web. En cada uno de ellos se generará en una única ocasión las clases de acceso al servicio y a partir de ahí a través de su interfaz permitirán a los portlets correspondientes obtener la información sin haber generado otra vez las clases.

Como se puede observar, el último elemento que queda por comentar es el servicio web de la LFP. En un principio se pensó en que el portal mostrara la información de las estadísticas de la LFP pero después de buscar alguna forma de obtenerla se observó que la propia liga de fútbol no da facilidades para acceder a la misma. Esto puede ser porque hay empresas que se lucran con esta información mediante juegos tipo Comunio [41] con el que ya han tenido problemas e incluso intentos de cerrar la página web. Debido a esto y teniendo en cuenta que el libro utilizado para aprender Java incluye un capítulo sobre servicios web [42] donde se estudió tanto la manera de consumir como de distribuir información a través de servicios web basados en SOAP, se decidió crear un servicio web con la información de la LFP que se deseaba mostrar, manteniendo la información en la base de datos.

A partir de este punto, una vez descrita la arquitectura lógica del sistema de forma general, se van a ir explicando tanto la lógica como el diseño estático y dinámico de cada uno de los componentes comentados: el servicio Web, los dos EJBs de sesión y los diferentes portlets a desarrollar.

5.4 Diseño de componentes

5.4.1 Servicio Web de estadísticas LFP

Como se ha comentado en el apartado de la arquitectura lógica, este componente es un servicio web local que ofrecerá información sobre estadísticas de la liga de fútbol profesional. La lógica de este componente se muestra en el siguiente diagrama:

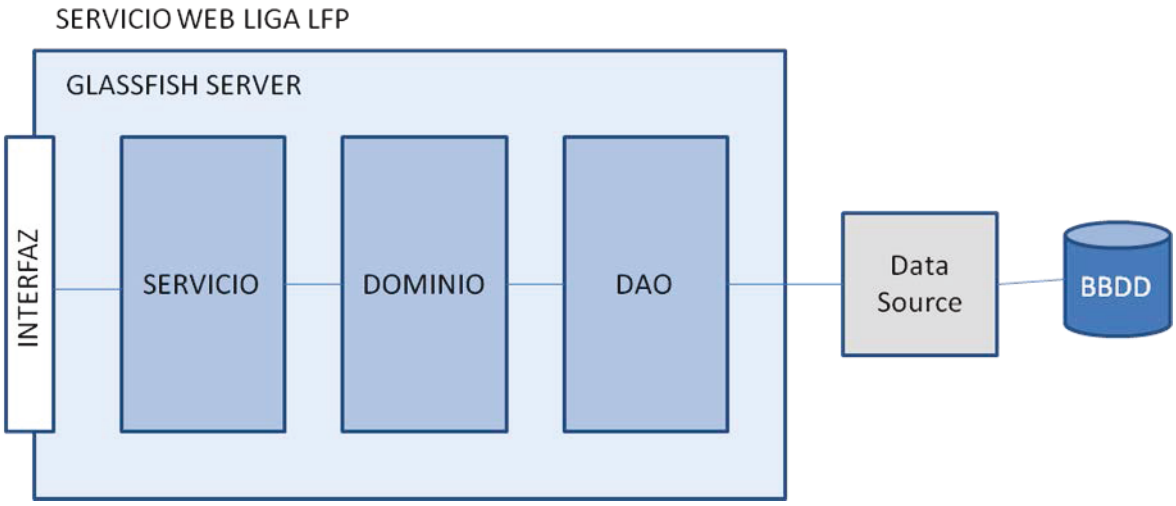

Ilustración 14. Arquitectura lógica servicio web de estadísticas LFP.

El servicio está diseñado basándose en una arquitectura de tres capas: la capa de servicio, que gestionará las peticiones entrantes al servicio web, la capa de DAO, que suministrará la interfaz para aislar la lógica del negocio del acceso a la base de datos y la capa de dominio, que contendrá clases auxiliares para el intercambio de datos entre las otras dos capas, pasando objetos completos en lugar de parámetros separados.

Un servicio web basado en SOAP siempre posee una interfaz con los métodos de acceso y una clase que implementa el funcionamiento de dichos métodos. Esta clase de implementación de los métodos de la interfaz es la clase que se encuentra en la capa de servicio.

A continuación se mostrarán cada una de las clases que forman parte de estas tres capas, junto con sus atributos y métodos y una descripción general de los mismos.

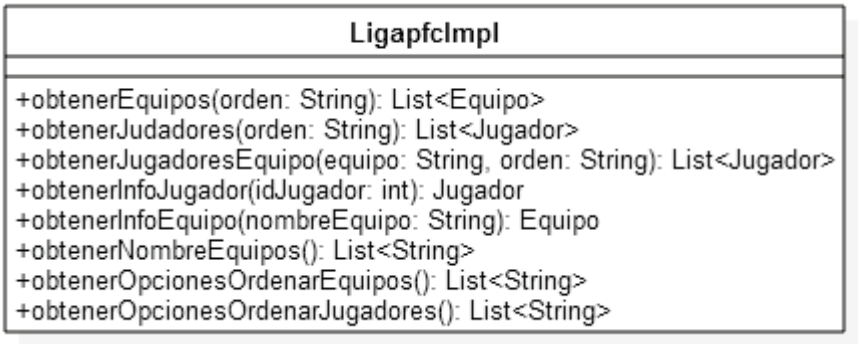

Ilustración 15. Clase LigapfcImpl.

Es la única clase de la capa de servicio y es la que gestiona todas las invocaciones que se realicen sobre el servicio web. No posee ningún atributo y mediante sus métodos se podrá obtener un listado de todos los equipos con el orden deseado, un listado de todos los jugadores, un listado de todos los jugadores de un equipo, ambos también ordenados según se desee, obtener la información de un equipo y un jugador concretos, un listado

con los nombres de todos los equipos y las opciones de ordenación tanto para los listados de jugadores como de equipos.

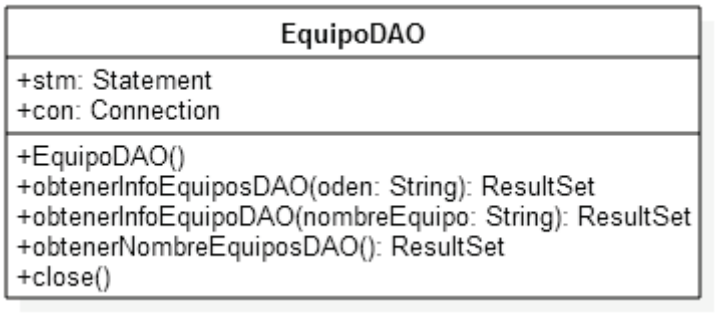

Ilustración 16. Clase EquipoDAO.

Esta clase contiene funciones para el acceso a la base de datos en referencia a los equipos. De esta forma se separa la lógica del negocio implementada en la clase de la capa de servicio de la tecnología de persistencia de la base de datos.

Posee dos atributos utilizados para realizar la conexión con la base de datos y como métodos el constructor donde se realiza la conexión, un método para cerrarla y tres métodos para obtener información: uno para obtener información sobre todos los equipos, otro para obtener la información de un equipo concreto y el último para obtener la lista de los nombres de todos los equipos.

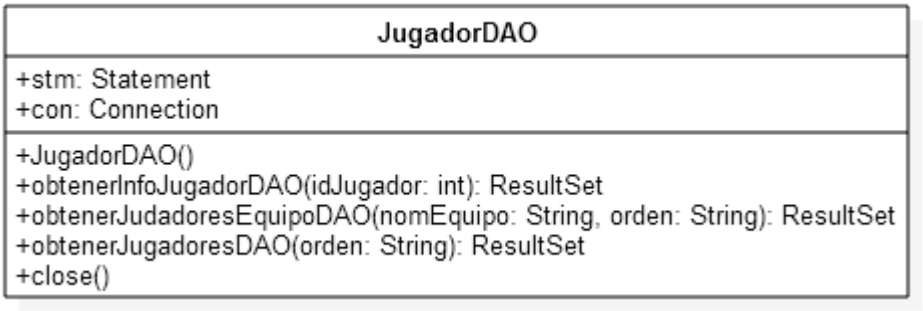

Ilustración 17. Clase JugadorDAO.

Esta clase contiene funciones para el acceso a la base de datos en referencia a los jugadores. Al igual que la anterior, separa la lógica del negocio implementada en la clase de la capa de servicio de la tecnología de persistencia de la base de datos.

Posee dos atributos utilizados para conectar con la base de datos. Posee tres métodos a parte del constructor y el método *close* para crear y cerrar la conexión con la base de datos. Estos métodos permiten obtener todos los datos de un jugador concreto, obtener toda la información de todos los jugadores y obtener todos los jugadores de un equipo concreto.

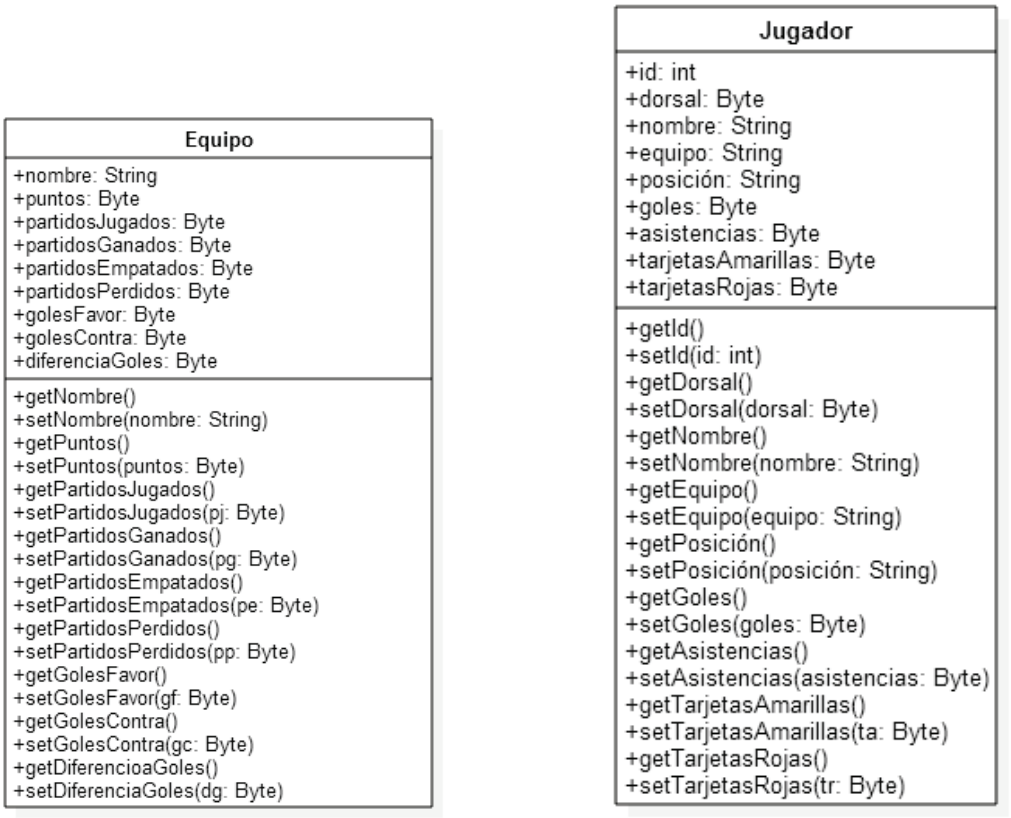

Ilustración 18. Clases Equipo y Jugador.

Estas clases forman parte de la capa de dominio. Contienen todos los atributos que poseen un equipo y un jugador en las tablas de la base de datos. Como métodos únicamente los get() y los set() de cada atributo. Tanto esta clase como la clase Equipo son utilizadas por la clase de la capa de servicio para devolver los datos pedidos por los clientes.

Cuando llega una invocación a un método del servicio, éste utiliza las clases correspondientes de la capa de DAO para obtener la información de la base de datos y las clases de la capa de dominio para devolver los resultados. De esta forma las clases de la capa de dominio no tienen relación con las clases de le la capa de DAO. Simplemente la clase implementadora del servicio es la que tiene relación con las clases de las dos capas, encargándose de gestionar la obtención del resultado requerido. Esto puede observarse en el siguiente diagrama de clases.

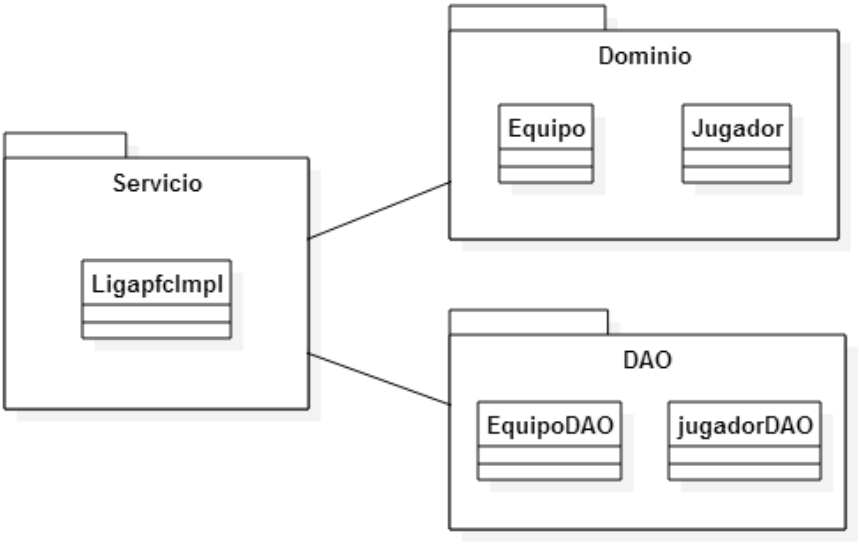

Ilustración 19. Diagrama de clases del servicio web LFP.

Una vez descrita la lógica del componente y definida la estructura de las clases se mostrará un diagrama dinámico de secuencia por cada método ofrecido por el servicio. Este tipo de diagrama muestra cómo interactúan los diferentes objetos en un sistema por lo que utilizándolos en conjunto con el diseño estático, ofrecerán una idea bastante clara del funcionamiento que tendrá este componente.

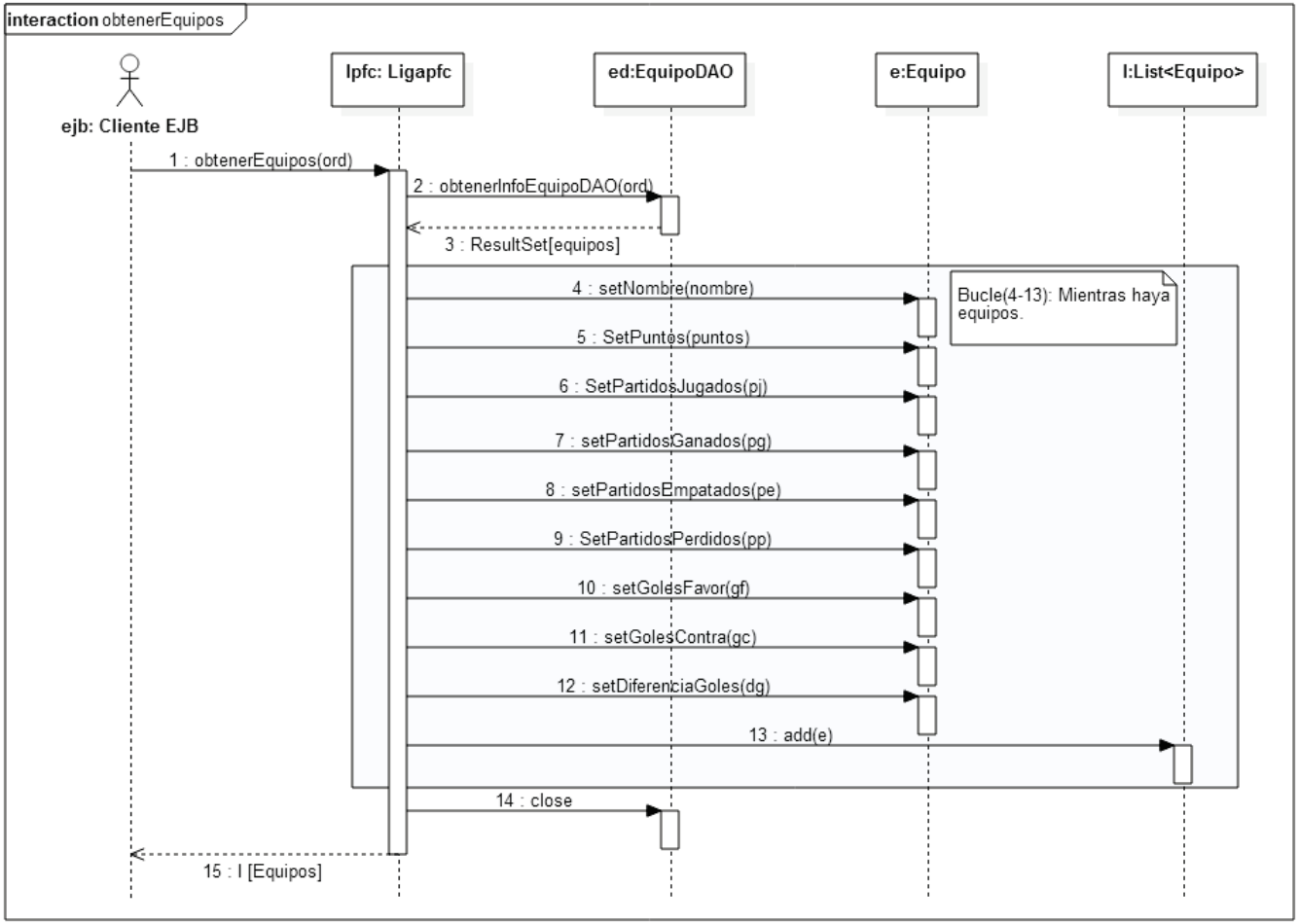

Ilustración 20. Diagrama de secuencia obtenerEquipos.

Para obtener todos los equipos, el método recoge toda la información de todos los equipos de la base de datos. Posteriormente mediante un bucle va rellenando objetos de la clase Equipo de la capa de dominio y los introduce en una lista, la cual será el resultado devuelto.

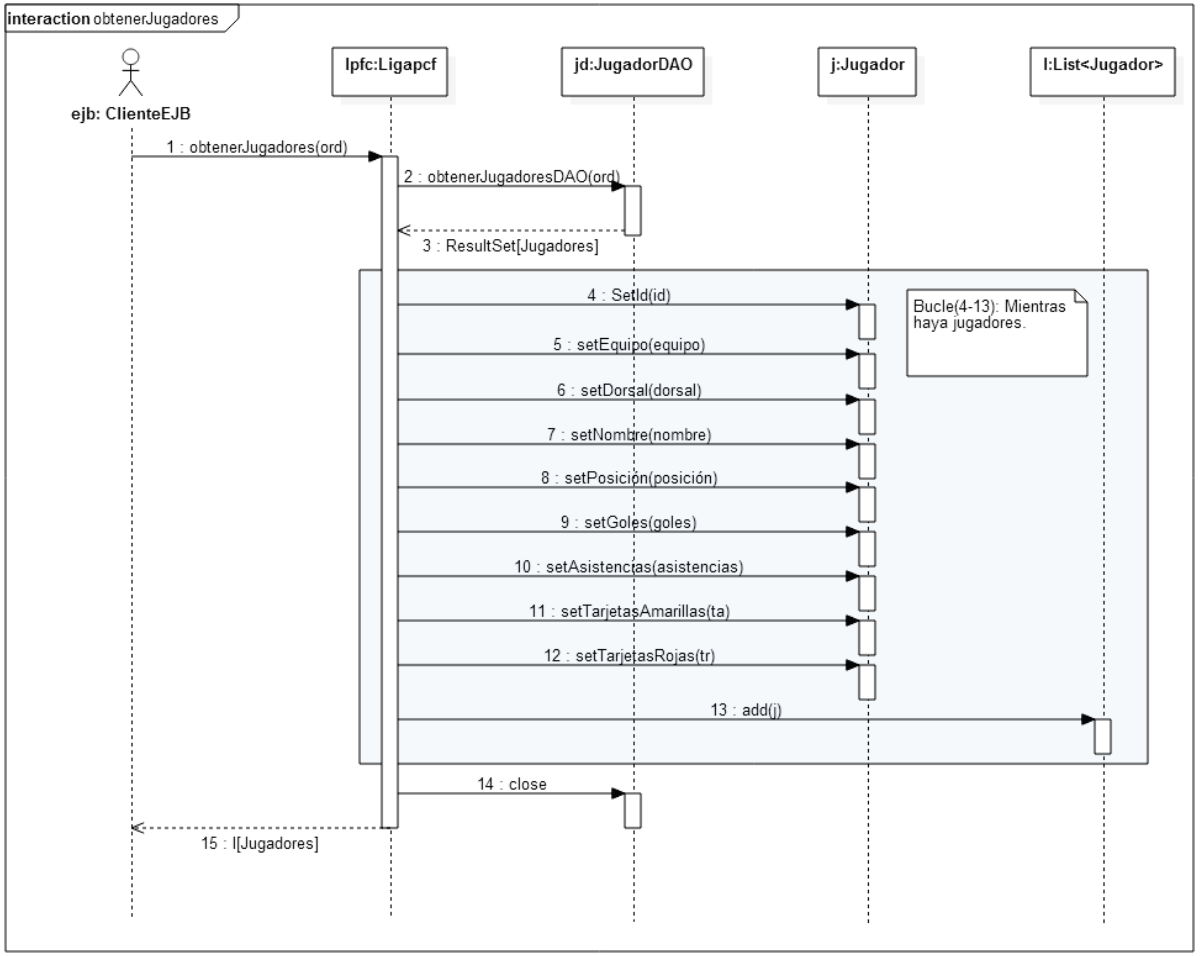

Ilustración 21. Diagrama de secuencia obtenerJugadores.

Para obtener todos los jugadores, el método recoge toda la información de todos los jugadores de la base de datos. Posteriormente mediante un bucle va rellenando objetos de la clase Jugador de la capa de dominio y los introduce en una lista, la cual será el resultado devuelto.

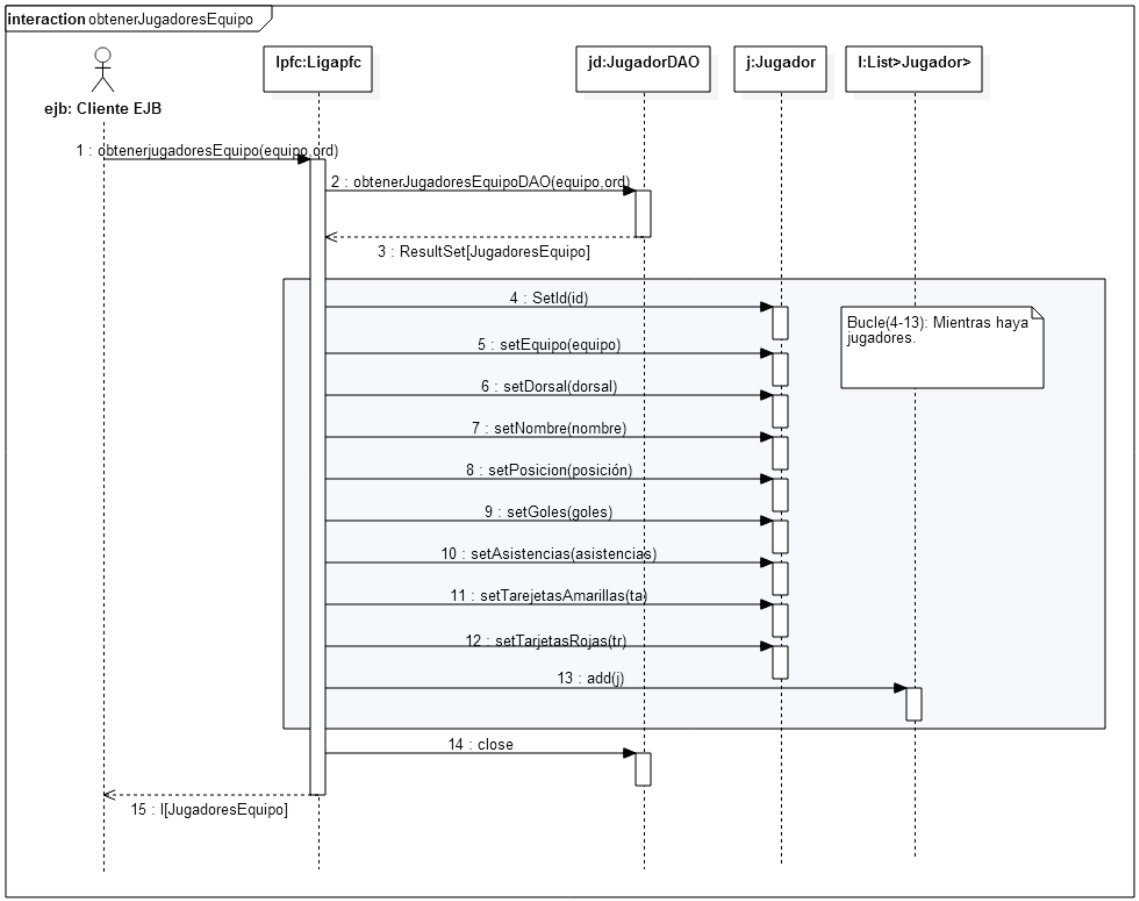

Ilustración 22. Diagrama de secuencia obtenerJugadoresEquipo.

Para obtener todos los jugadores de un equipo, el método recoge toda la información de todos los jugadores del equipo pasado como parámetro de la base de datos. Posteriormente mediante un bucle va rellenando objetos de la clase Jugador de la capa de dominio y los introduce en una lista, la cual será el resultado devuelto.

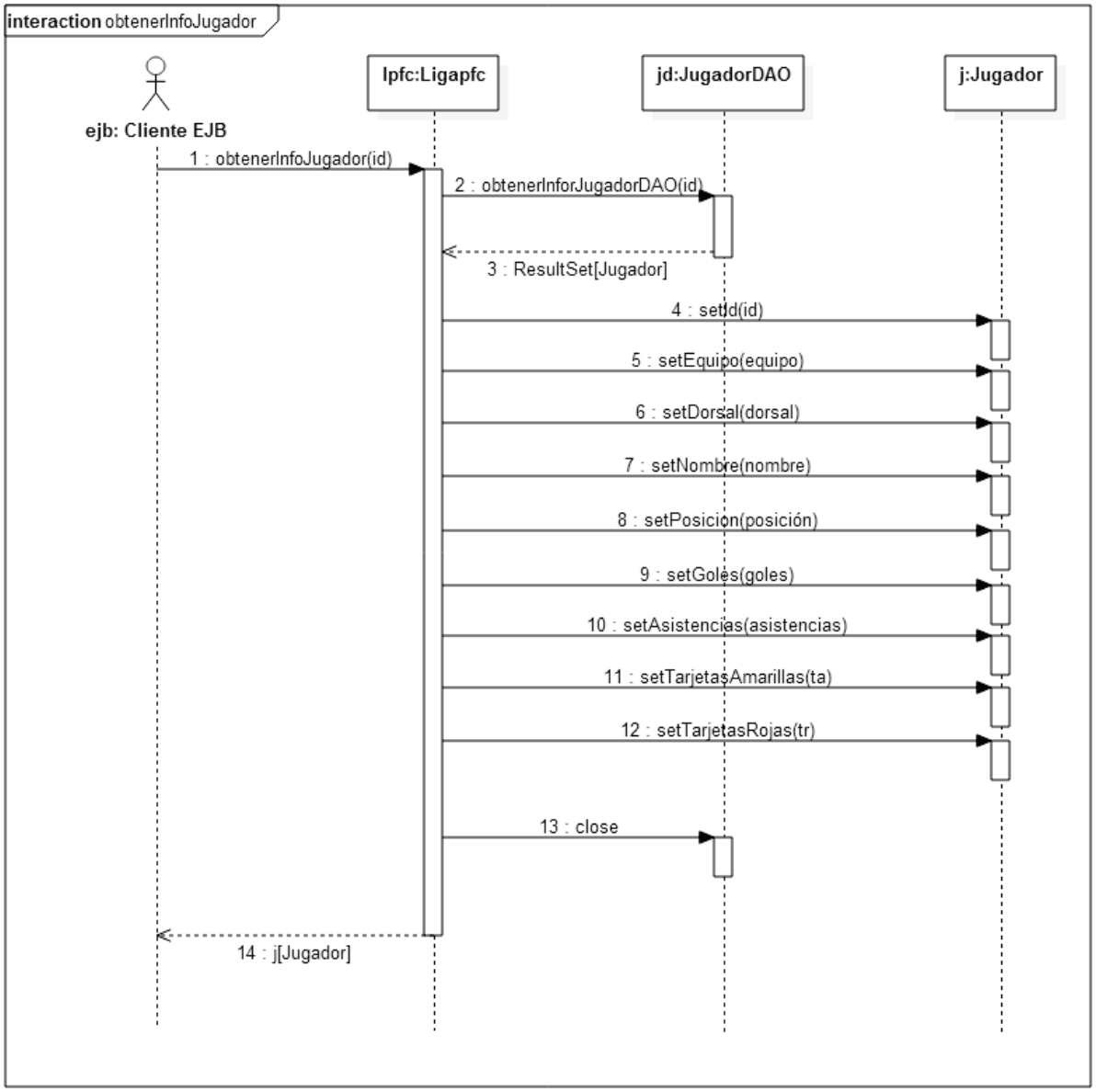

Ilustración 23. Diagrama de secuencia obtenerInfoJugador.

Para obtener toda la información de un jugador, el método recoge toda la información del jugador pasado como parámetro de la base de datos. Posteriormente se rellena un objeto de la clase Jugador de la capa de dominio y se devuelve al cliente como resultado de la operación.

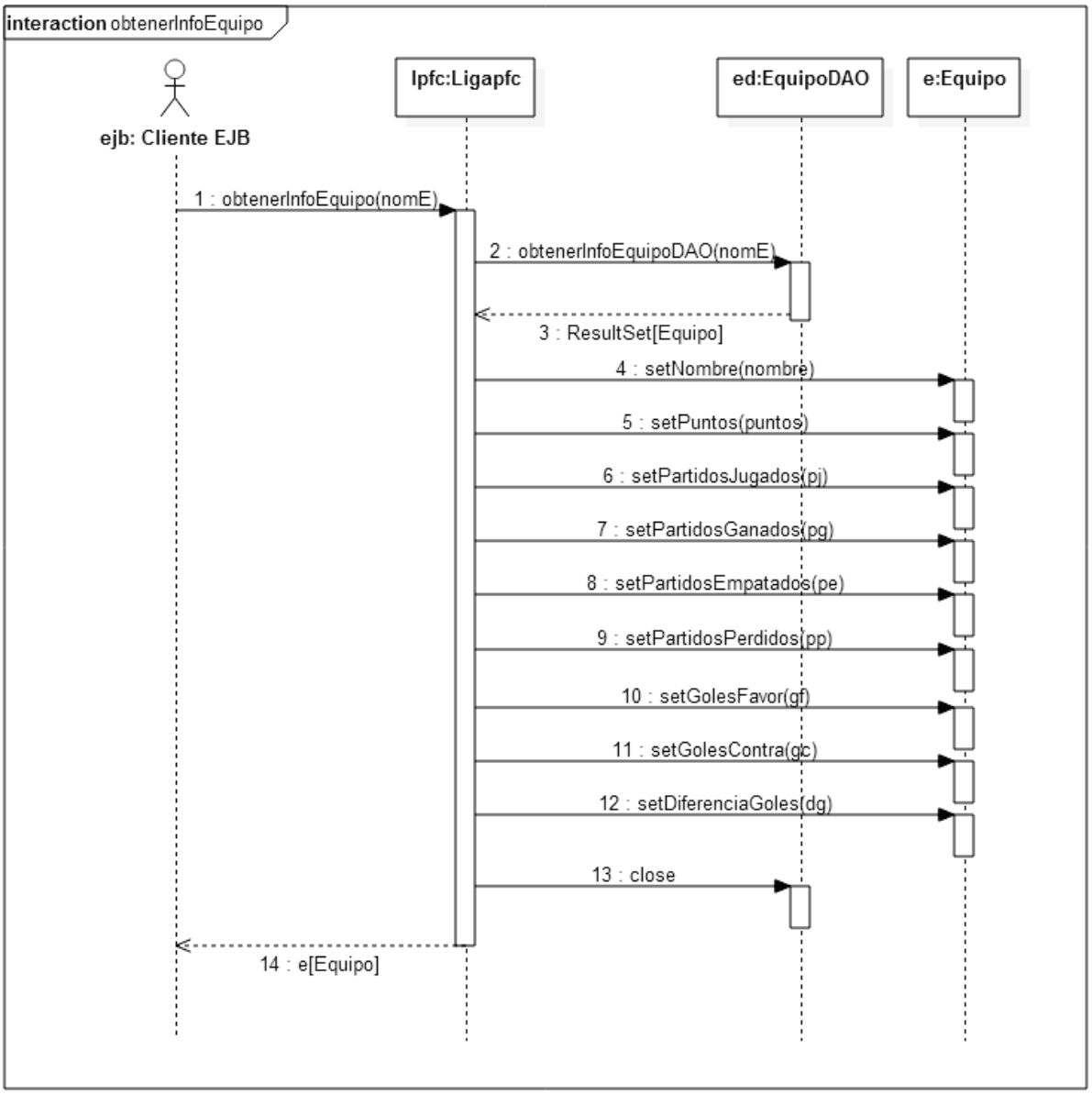

Ilustración 24. Diagrama de secuencia obtenerInfoEquipo.

Para obtener toda la información de un equipo, el método recoge toda la información del equipo pasado como parámetro de la base de datos. Posteriormente se rellena un objeto de la clase Equipo de la capa de dominio y se devuelve al cliente como resultado de la operación.

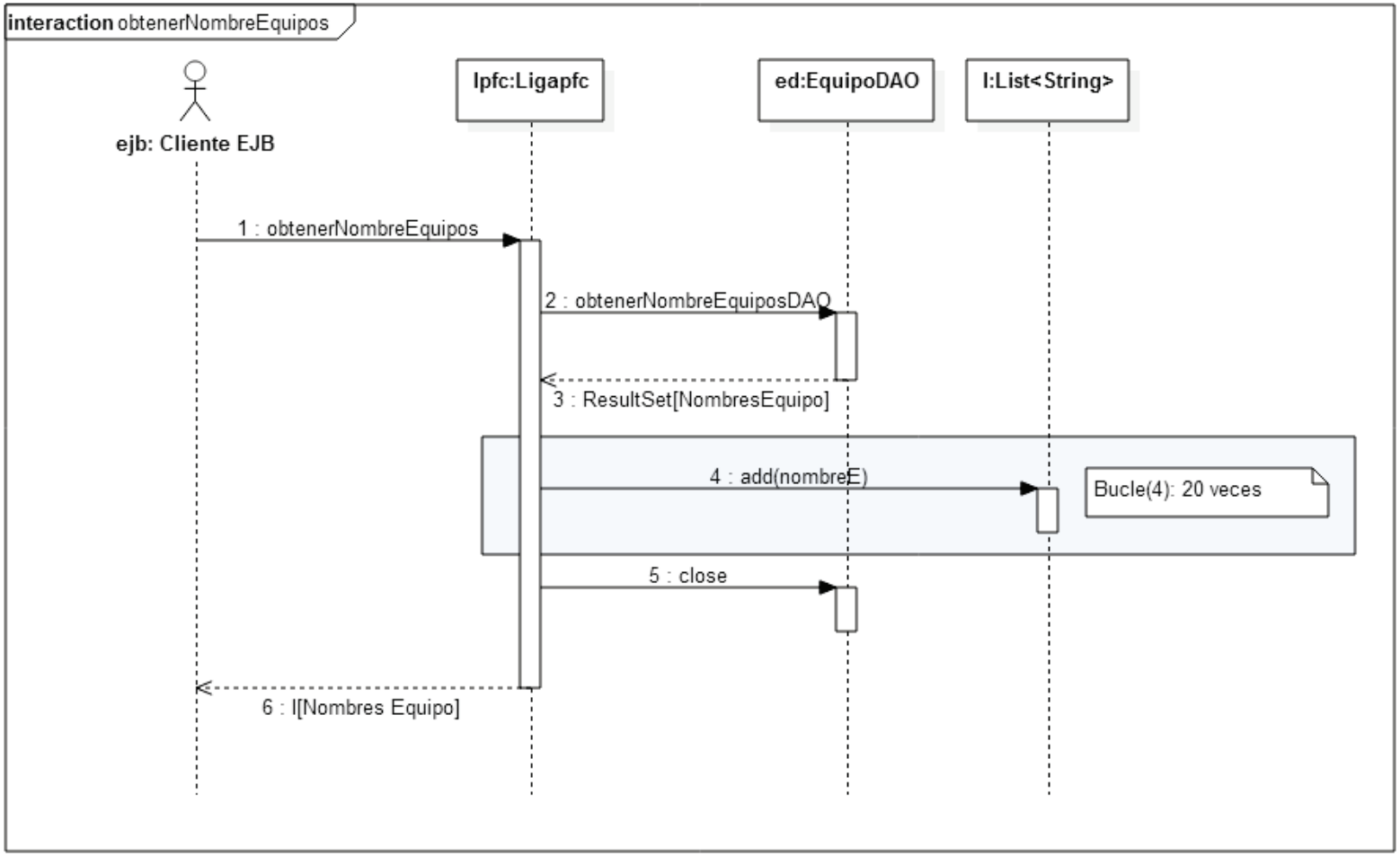

Ilustración 25. Diagrama de secuencia obtenerNombreEquipos.

Para obtener los nombres de todos los equipos, el método llama a la función de la capa DAO que realiza la query que obtiene únicamente los nombres de los equipos de la base de datos. Posteriormente mediante un bucle los introduce en una lista, la cual será el resultado devuelto.

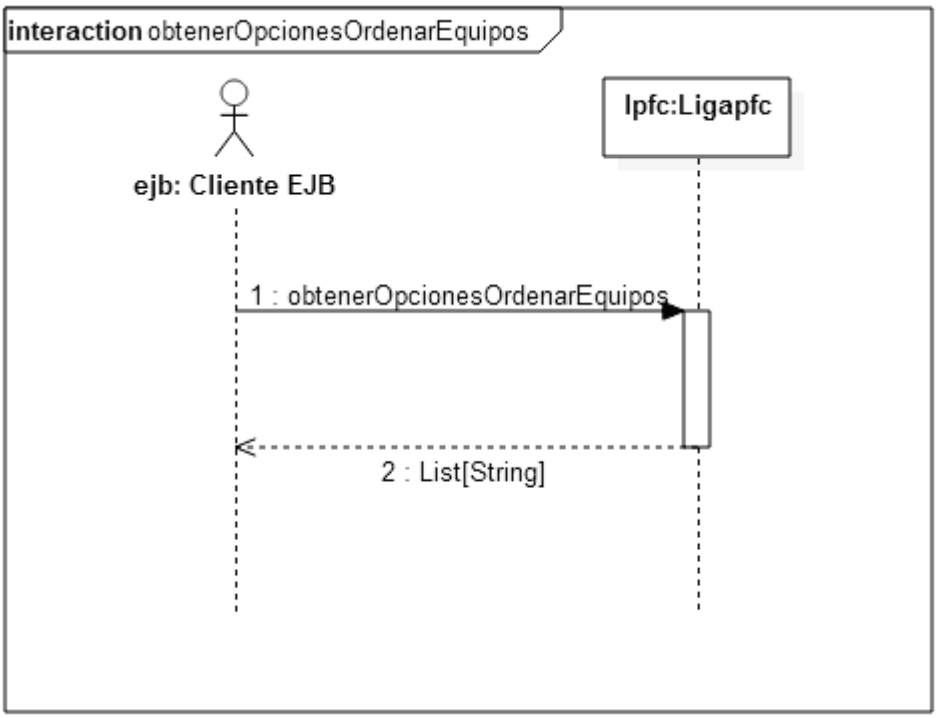

Ilustración 26. Diagrama de secuencia obtenerOpcionesOrdenarEquipos.

Para obtener las opciones de ordenación de equipos simplemente se devuelve una lista ya que estas son fijas.

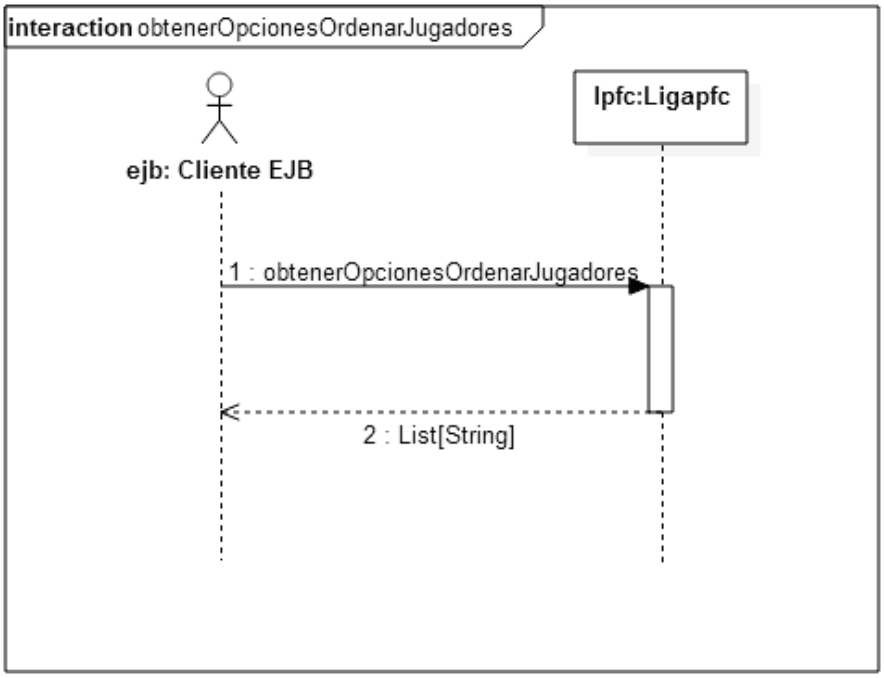

Ilustración 27. Diagrama de secuencia obtenerOpcionesOrdenarJugadores.

Para obtener las opciones de ordenación de jugadores simplemente se devuelve una lista ya que estas son fijas.

5.4.2 EJB cliente del Servicio Web LFP

En el diagrama que se muestra a continuación se muestra la lógica del componente EJB que hará de cliente del servicio web de la LFP.

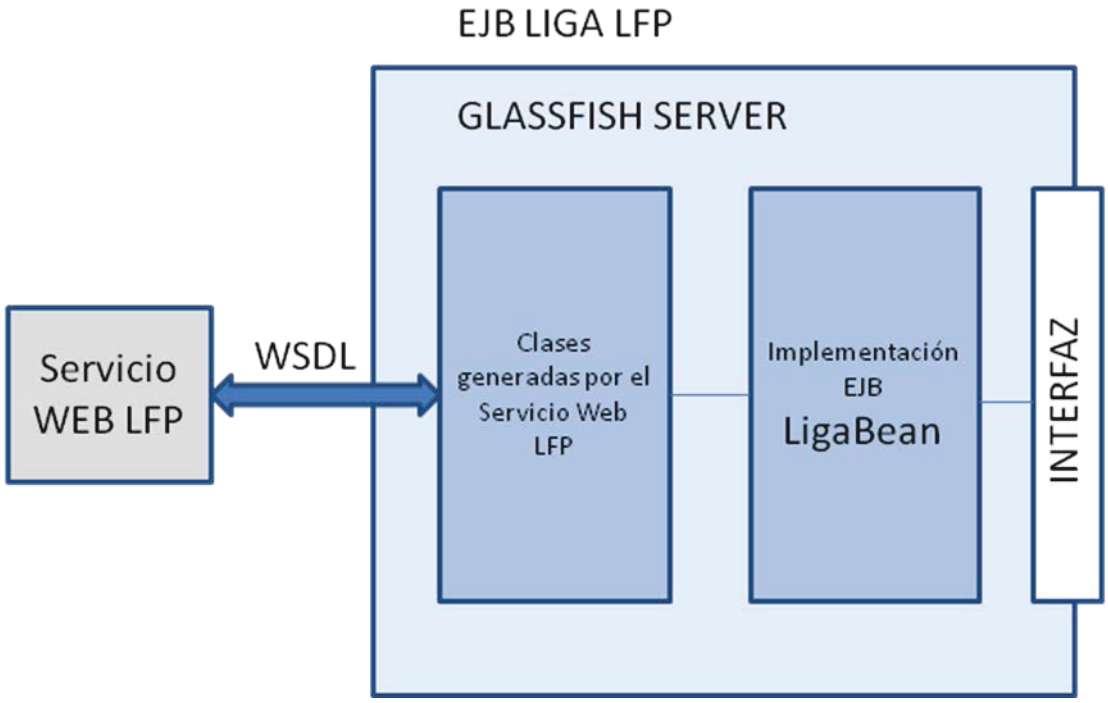

Ilustración 28. Arquitectura lógica EJB cliente servicio web LFP.

Como ya se comento en el apartado tecnologías cuando se comentaron los EJB, todo módulo EJB tiene una interfaz con el que los usuarios del mismo podrán acceder a él y una clase que implementa las funciones ofrecidas en la interfaz. Aparte de estas, el módulo contendrá las clases generadas del servicio web a partir de su descriptor WSDL, con las que se podrá conectar y acceder a la información del mismo. Este módulo, al hacer de cliente del servicio web se encargará de obtener la información que los portlets mostrarían en caso de ser clientes directos del servicio.

La clase de implementación del componente EJB de la Liga de fútbol se definirá de la siguiente forma:

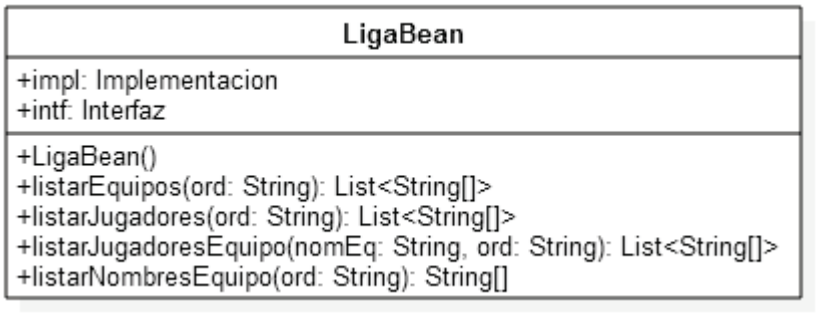

Ilustración 29. Clase LigaBean.

La clase posee dos atributos que son necesarios para poder acceder a la información del servicio web. El primero será un objeto de la clase de implementación del servicio web que ha sido generada junto con el resto de clases del servicio mediante el WSDL. El segundo es de tipo interfaz que fue definida durante la creación del servicio web. Ambos en conjunto se utilizarán para conectar al servicio web y obtener la información. La conexión se realiza en el constructor de la clase.

Aparte del constructor el EJB de la LFP ofrecerá métodos para obtener un listado de los equipos con sus estadísticas, un listado de todos los jugadores, un listado de los jugadores de un equipo concreto y por último un listado de los nombres de los equipos únicamente. Todos los listados podrán ser obtenidos ordenados por un campo específico.

La clase de implementación del EJB llamada LigaBean.java está relacionada obligatoriamente con las clases implementación y la interfaz generadas del servicio web ya que estas dos clases son necesarias para conectar con el servicio web y acceder a la información. El resto de clases generadas únicamente estarán relacionadas con la primera en el caso de que sean requeridas para obtener cierta información. Esto es así debido a que el servicio web puede ofrecer información no mostrada en los portlets, pero podría ocurrir que otra implementación de un cliente del servicio si deseara acceder a esa información.

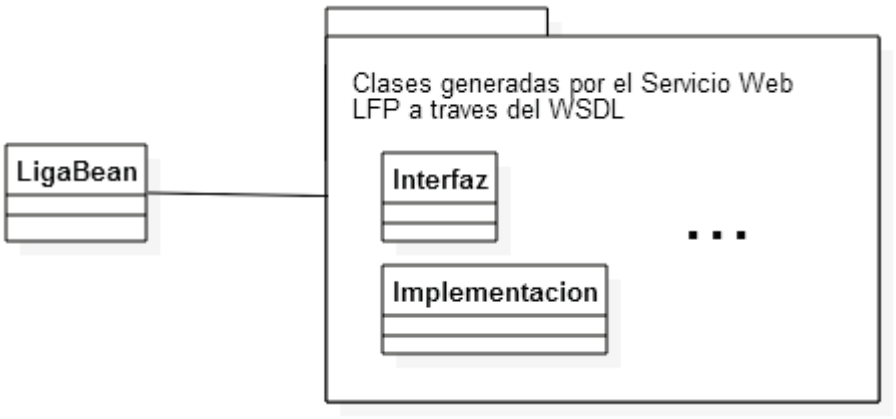

Ilustración 30. Diagrama de clases EJB cliente servicio web LFP.

A continuación se mostrarán diferentes diagramas dinámicos de secuencia para los métodos ofrecidos por el módulo a través de su interfaz. Utilizándolos en conjunto con el diseño estático, ofrecerán una idea bastante clara del funcionamiento que tendrá este componente.

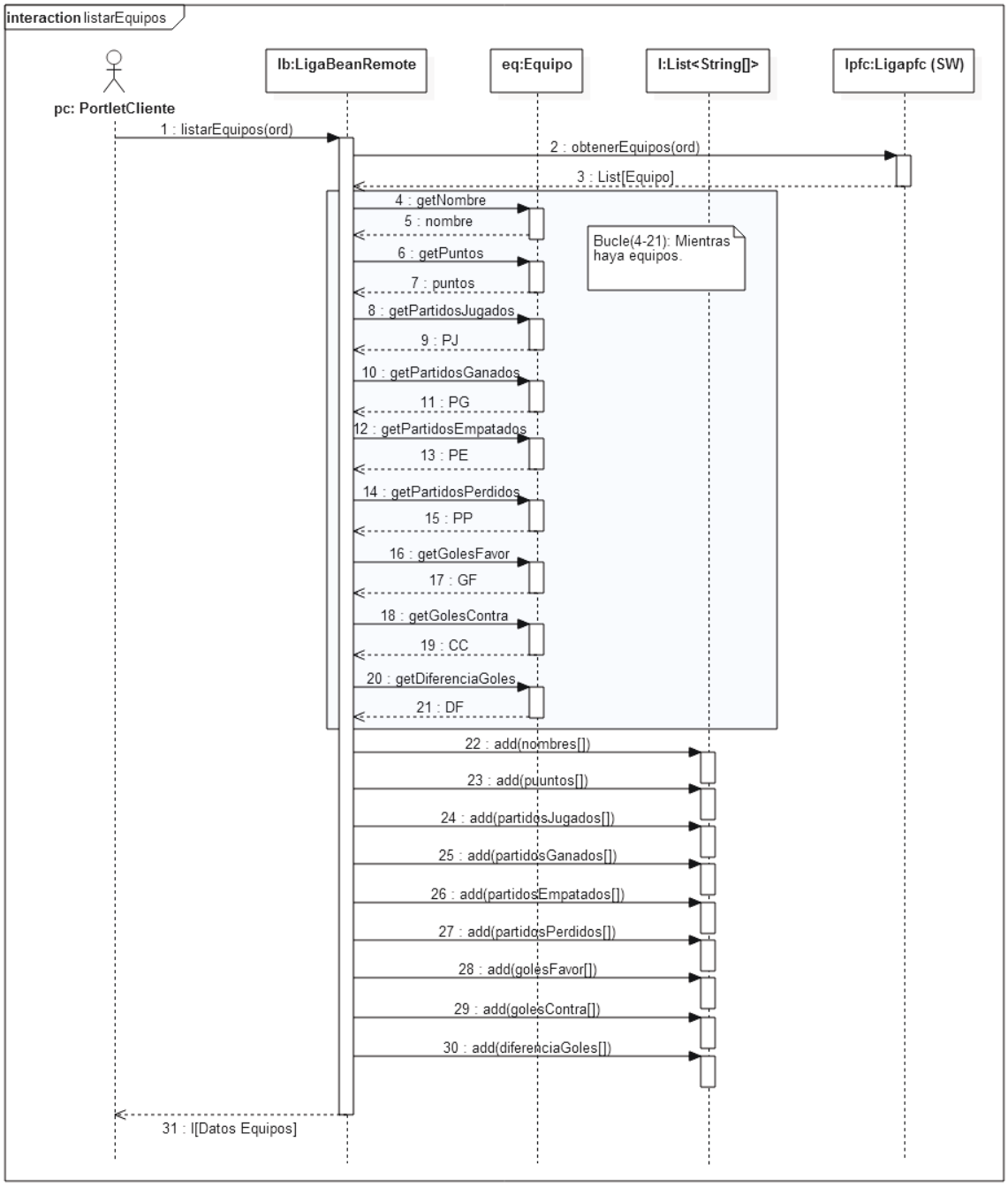

Ilustración 31. Diagrama de secuencia listarEquipos.

Esta operación para obtener la información invoca al método del servicio web *obtenerEquipos* que le devuelve un listado de los equipos. Mediante un bucle se van introduciendo cada atributo de cada equipo en un array con el mismo nombre del atributo. De esta manera se devolverá el mismo atributo de cada equipo en un mismo array, lo que hace más fácil la disposición posterior en el portlet.

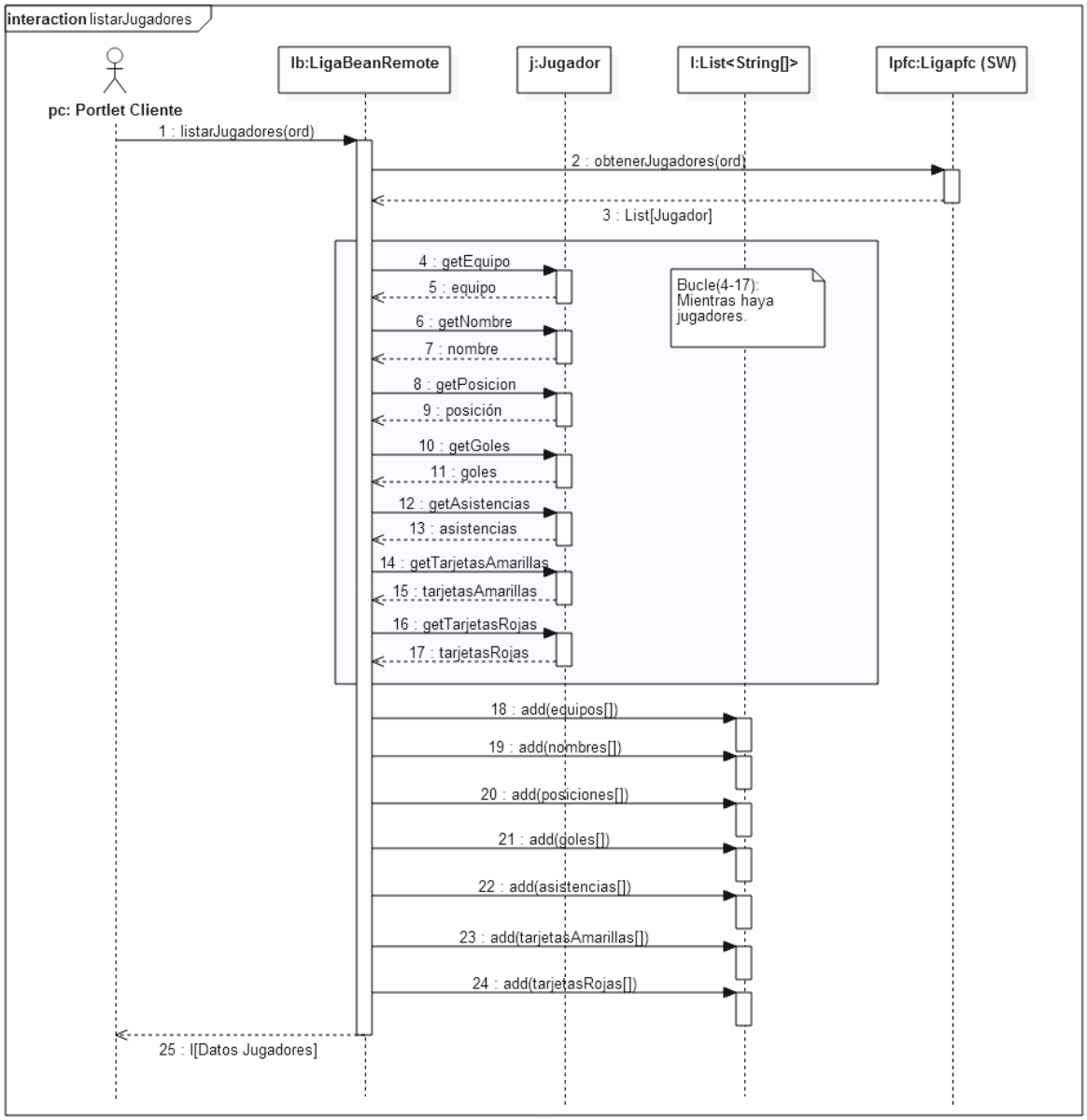

Ilustración 32. Diagrama de secuencia listarJugadores.

Esta operación para obtener la información invoca al método del servicio web *obtenerJugadores* que le devuelve un listado de los jugadores. Al igual que el diagrama de secuencia anterior, mediante un bucle se va introduciendo cada atributo de cada jugador en un array con el mismo nombre para devolverlos todos juntos y facilitar posteriormente su colocación en el portlet.

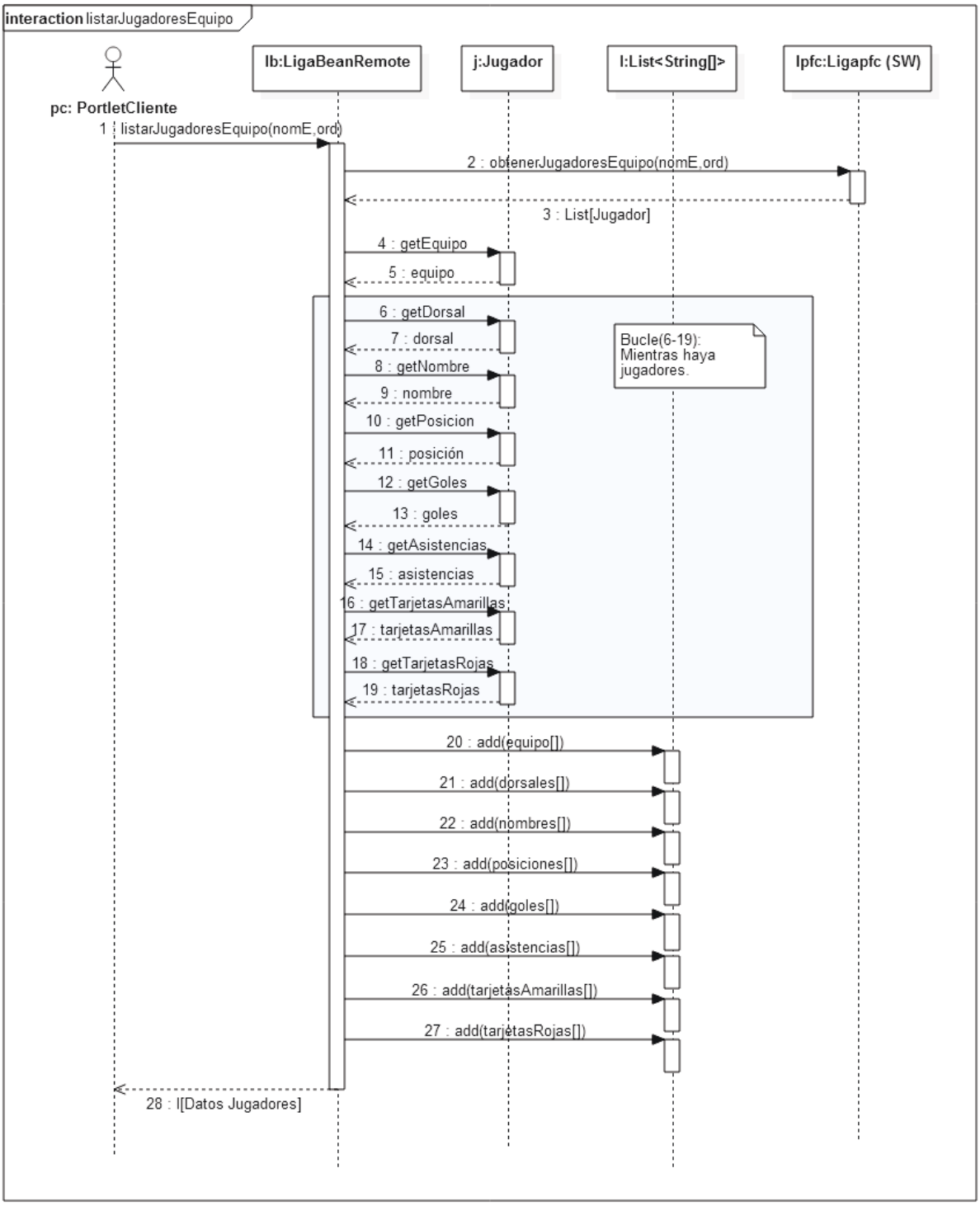

Ilustración 33. Diagrama de secuencia listarJugadoresEquipo.

Este método es igual que el método *listarJugadores* salvo porque se invoca el método *obtenerJugadoresEquipo* del servicio web que únicamente devuelve el listado de jugadores de un equipo. A partir de ahí el listado se trata de la misma manera: juntando todos los atributos iguales en un array con el nombre del atributo para devolverlos juntos.

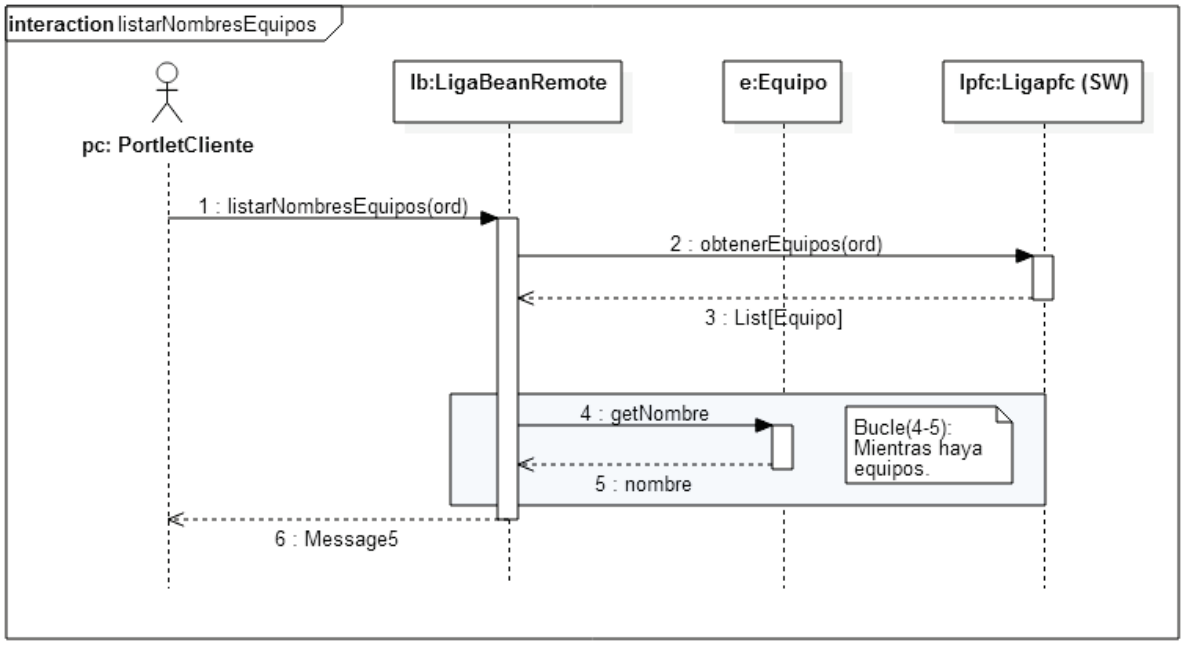

Ilustración 34. Diagrama de secuencia listarNombresEquipos.

Para obtener el listado de los nombres de los equipos se invoca el método del servicio web *obtenerEquipos* que devuelve un listado de todos los equipos. Posteriormente, mediante un bucle se obtiene únicamente el nombre de cada equipo y se devuelven todos en un array.

5.4.3 EJB cliente del Servicio Web Brasil 2014

En el siguiente diagrama se puede ver la lógica del componente EJB que hará de cliente del servicio web del Mundial de fútbol de Brasil 2014.

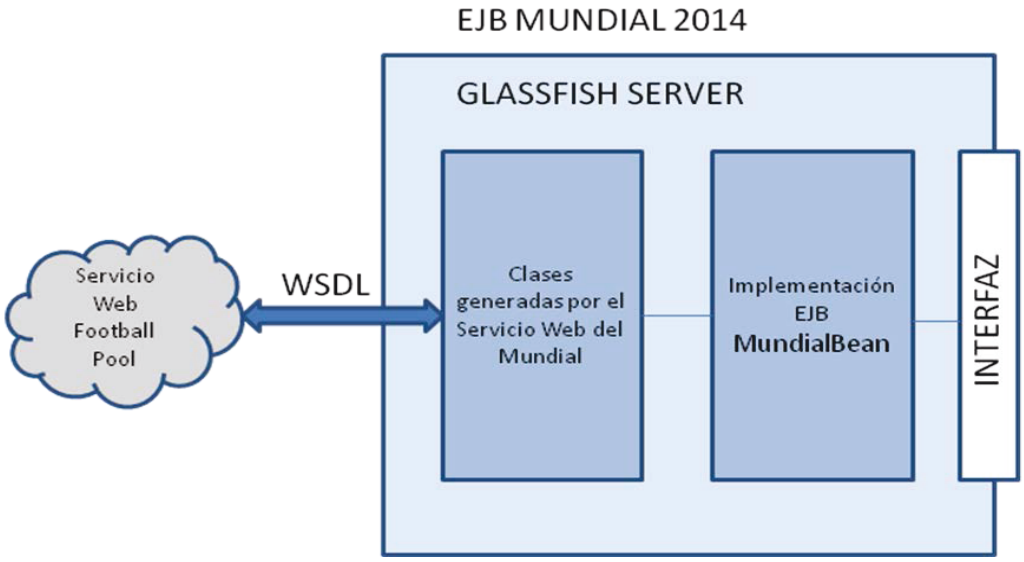

Ilustración 35. Arquitectura lógica EJB cliente servicio web mundial 2014.

Este módulo EJB ofrece una interfaz a los portlets que van a mostrar la información del mundial. En este módulo se van a generar las clases del servicio web a partir del documento descriptor WSDL con las cuales se podrá acceder a las funcionalidades del servicio. En la clase principal del EJB se implementarán todas las operaciones declaradas en la interfaz del mismo y que ofrecerán a los portlets toda la información que ellos mismos obtendrían siendo clientes directos del servicio web.

La clase de implementación del módulo EJB del mundial 2014 será definida de la siguiente forma:

Ilustración 36. Clase MundialBean.

Esta clase, como ya se comentó en el anterior componente EJB de la LFP tiene dos atributos, el primero del tipo de la clase generada que implementa los métodos ofrecidos por el servicio web y el segundo del tipo de la interfaz del servicio, también generada a partir del WSDL. Ambos se utilizan para poder conectar con se servicio web. Una vez realizada la conexión, mediante el atributo interfaz se podrá acceder a todos los métodos del servicio y utilizarlos para obtener la información deseada y generar la implementación de los métodos del propio módulo EJB.

Los métodos que ofrecerá el EJB incluyen, además del constructor donde se realiza la ya comentada conexión con el servicio web, otros métodos que permitirán obtener los 10 máximos goleadores del mundial, obtener información básica de todas las selecciones, obtener las selecciones de todos los grupos del mundial, obtener la información de un estadio concreto y obtener la plantilla completa de una selección completa.

En este componente la clase de implementación del EJB llamada MundialBean.java está relacionada obligatoriamente con las clases implementación y la interfaz generadas del servicio web ya que estas dos clases son necesarias para conectar con el servicio web y acceder a la información. El resto de clases generadas únicamente estarán relacionadas con la primera en el caso de que sean necesarias para obtener información. Esto ocurre porque el servicio web ofrece mucha más información de la que finalmente se va a utilizar, por lo tanto habrá clases generadas no utilizadas.

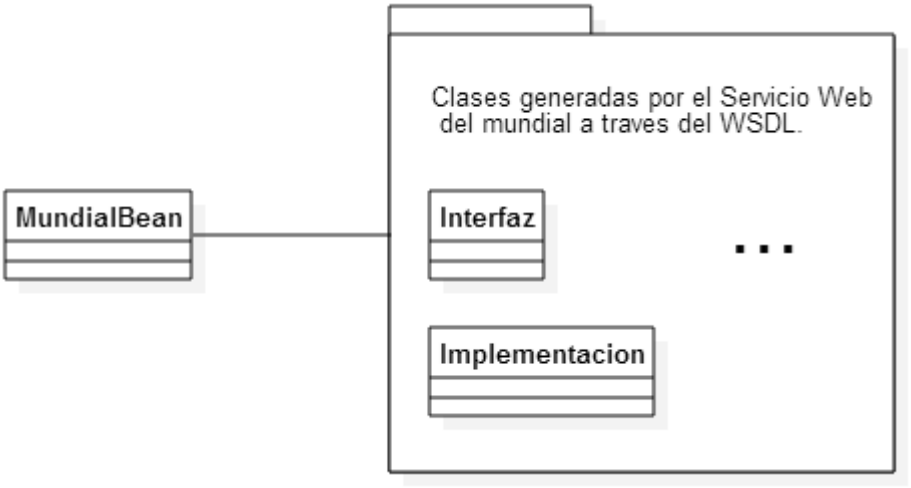

Ilustración 37. Diagrama de clases EJB cliente servicio web mundial 2014.

Una vez descrita la lógica del componente y definida la estructura de las clases se mostrará un diagrama dinámico de secuencia por cada método ofrecido por el EJB. Utilizándolos en conjunto con el diseño estático, ofrecerán una idea bastante clara del funcionamiento que tendrá este componente.

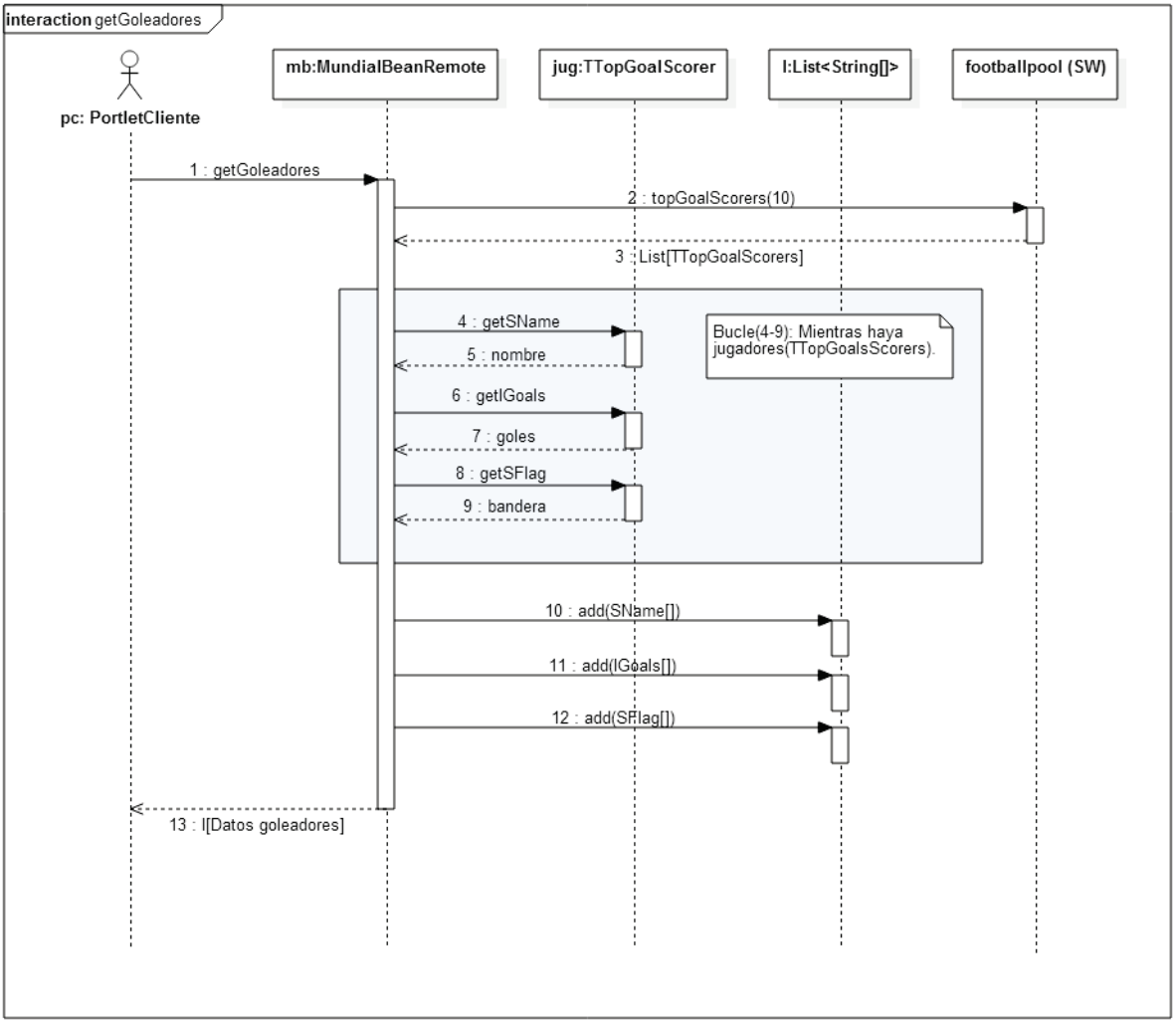

Ilustración 38. Diagrama de secuencia getGoleadores.

Para obtener la información de máximos goleadores se invoca el método correspondiente del servicio web indicándole la cantidad de jugadores a mostrar. Éste devuelve el listado de jugadores. Se obtienen los atributos de cada jugador y se introducen en arrays con el nombre del atributo para devolverlos todos juntos y facilitar su colocación en el portlet.

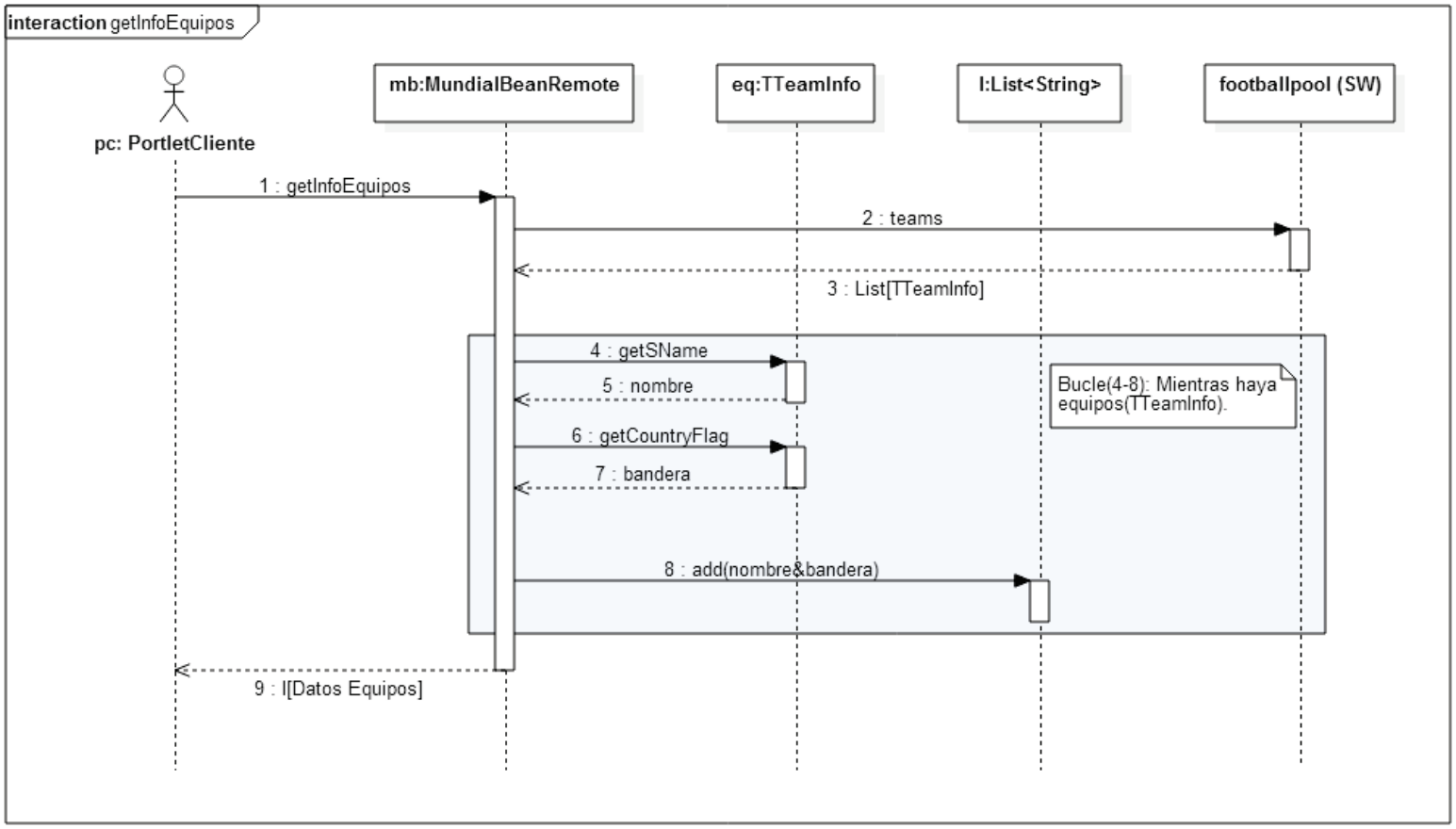

Ilustración 39. Diagrama de secuencia getInfoEquipos.

Para obtener la información de las selecciones se invoca el método correspondiente del servicio web que devuelve un listado con la información de las selecciones. Se obtienen los atributos deseados de cada selección y se concatenan en un String separados por caracteres de separación. En el portlet se separarán haciendo uso de la función *split* que separa cadenas de caracteres por el carácter indicado. Finalmente se devuelve una lista con los Strings de cada selección.

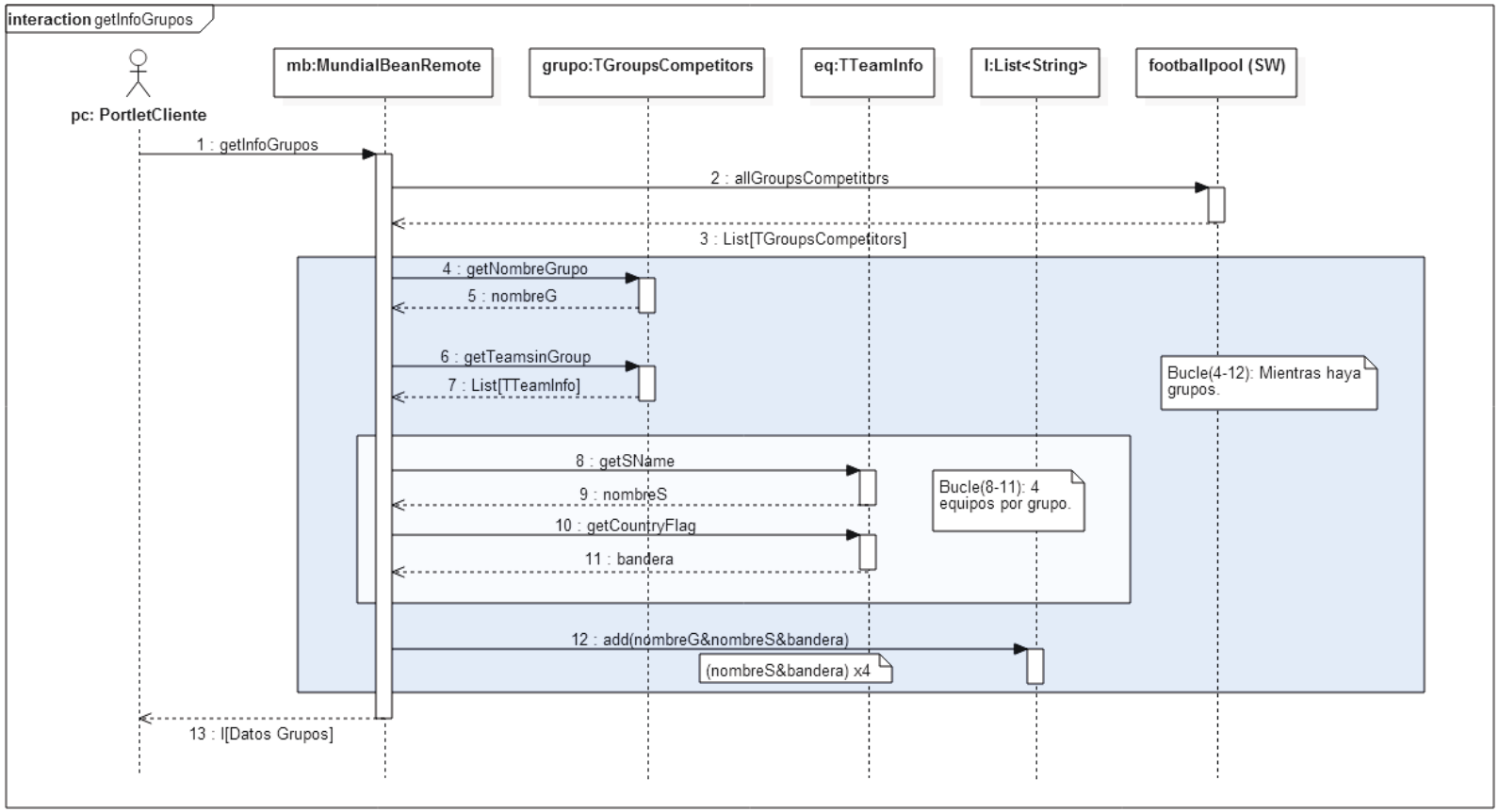

Ilustración 40. Diagrama de secuencia getInfoGrupos.

Para obtener la información de los grupos se invoca el método que devuelve un listado de todos los grupos. Posteriormente de cada grupo primero se obtiene el nombre del grupo y después se invoca el método que devuelve las selecciones de cada grupo. Se obtienen los atributos deseados de cada selección y se concatenan en un String junto con el nombre del grupo al que pertenecen. Finalmente se añaden todos los Strings en una lista y se devuelven. En el portlet se separarán mediante la función *split*.

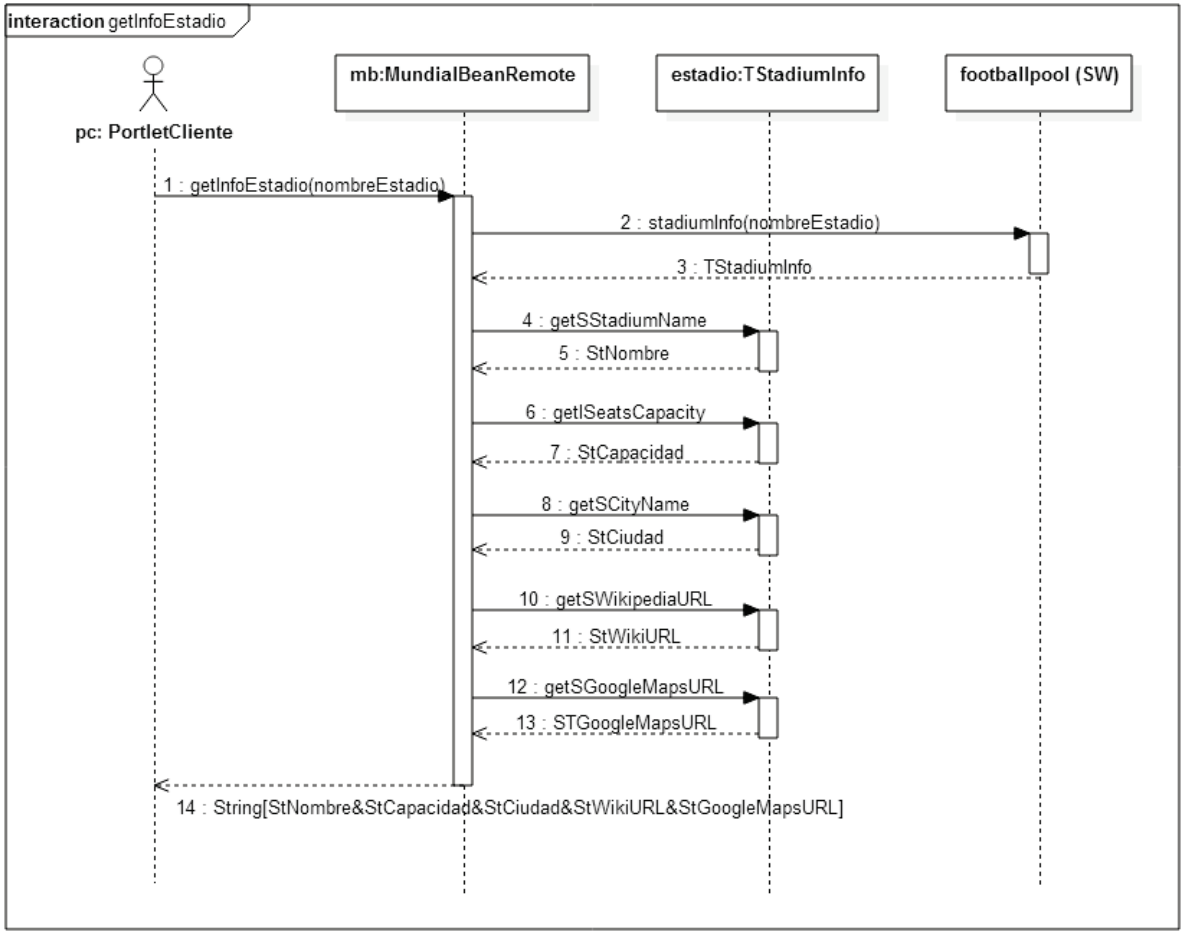

Ilustración 41. Diagrama de secuencia getInfoEstadio.

Para obtener la información de un estadio se invoca el método del servicio web que devuelve la información del estadio pasado como parámetro, se obtienen de ese estadio todos los atributos deseados y se concatena en un String, el cual se devuelve. En el portlet se separarán mediante la función *split*.

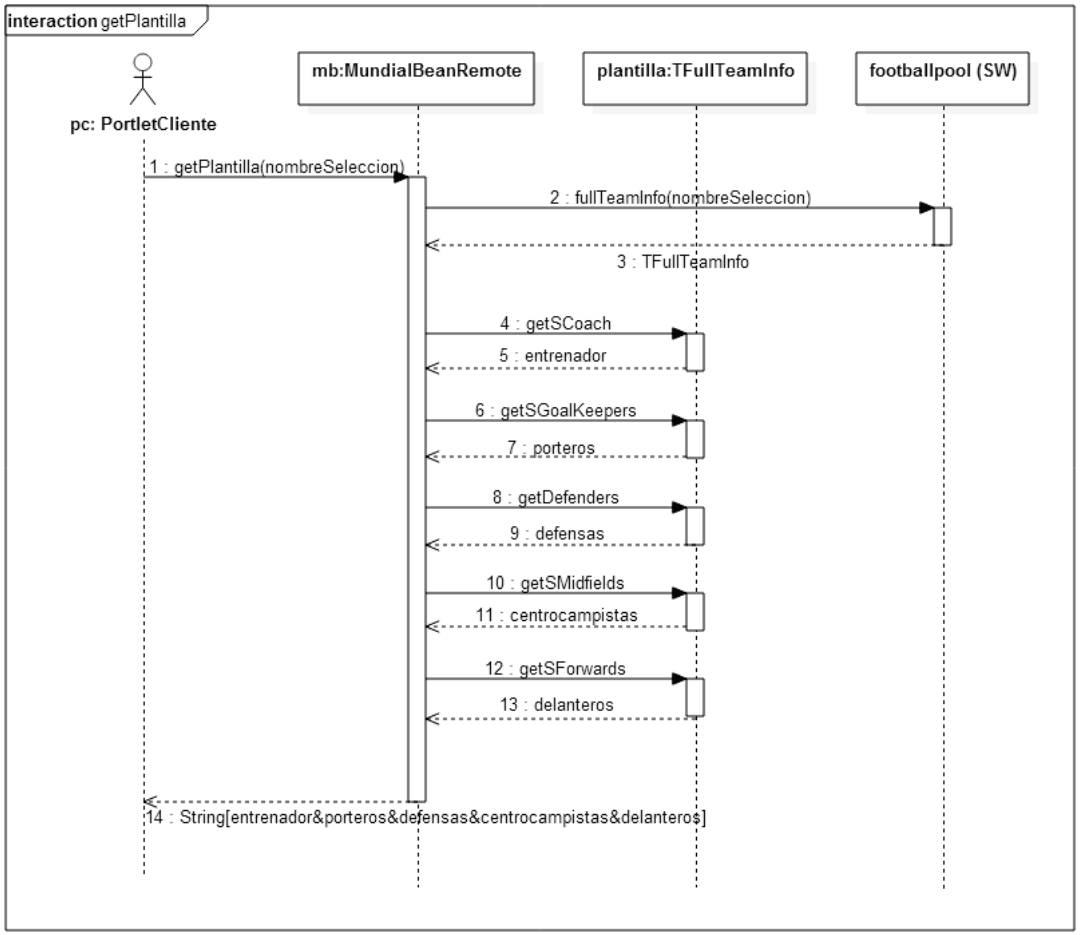

Ilustración 42. Diagrama de secuencia getPlantilla.

Para obtener todos los jugadores de una plantilla se invoca al método del servicio web que devuelve un objeto con toda la información de una selección. Sobre ese objeto se invocan los métodos que devuelven los jugadores separados por su posición. Todos ellos se concatenan en un mismo String pero separados por diferentes caracteres de separación diferenciando entre posiciones y dentro de cada posición entre jugadores.

5.4.4 Hook de Liferay listener de mensajes de Twitter.

El siguiente esquema muestra la lógica del hook de Liferay que se encargará de obtener los mensajes de Twitter y de introducirlos en la base de datos.

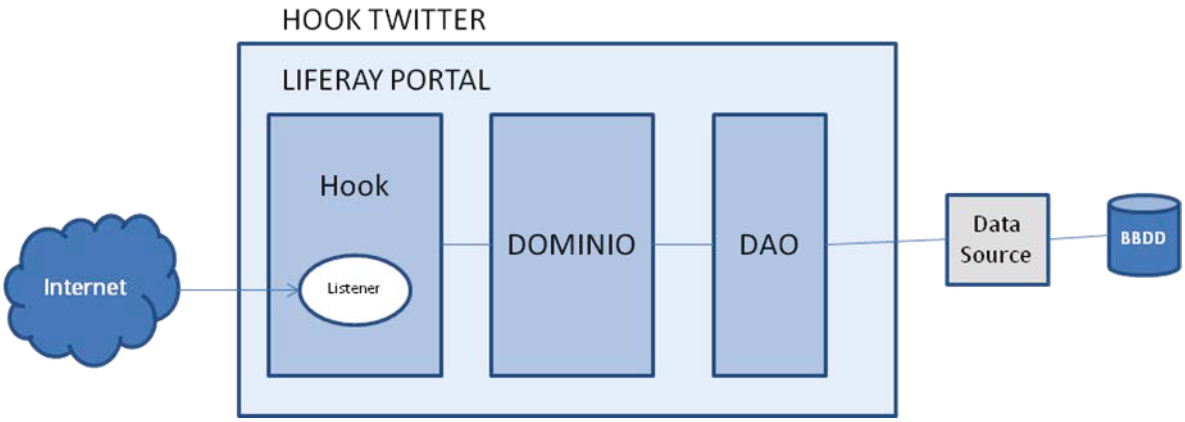

Ilustración 43. Arquitectura lógica del hook de Liferay.

Como es necesario que el hook acceda a la base de datos se ha diseñado basándose en la arquitectura de tres capas. Como ocurría en el servicio web de la LFP anteriormente explicado, se va a utilizar esta arquitectura para separar la lógica del negocio del acceso a la base de datos. Como ya se comentó en el servicio web, la capa de DAO suministra la interfaz para separar la implementación del Hook del acceso a la base de datos mientras que la capa de dominio contiene las clases auxiliares para el intercambio de datos entre el hook y la capa de DAO, pasando objetos completos en lugar de parámetros separados.

A continuación se mostrarán el diseño de cada una de las clases que forman parte de estas tres capas, junto con sus atributos y métodos y una descripción general de los mismos.

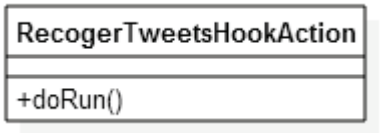

Ilustración 44. Clase RecogerTweetsAction.

La clase que implementa el funcionamiento del hook llamada *RecogerTweetsHookAction* únicamente posee un método que es ejecutado automáticamente nada más ser desplegado o cuando el servidor de Liferay es lanzado si este no estaba en funcionamiento.

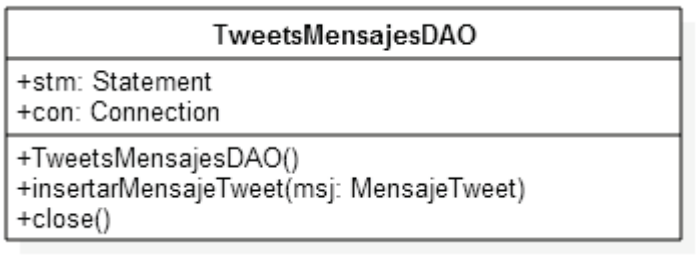

Ilustración 45. Clase TweetsMensajesDAO.

Esta clase es utilizada para introducir los mensajes o Tweet recogidos por el listener en la base de datos.

Posee dos atributos utilizados para realizar la conexión con la base de datos y como métodos el constructor donde se realiza la conexión, un método para cerrarla y un método con el que se introduce cada mensaje que llega en la base de datos.

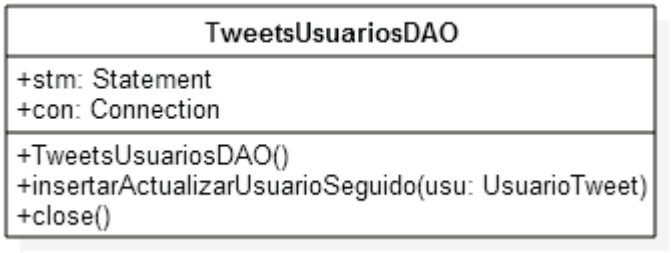

Ilustración 46. Clase TweetsUsuariosDAO.

Esta clase es utilizada para introducir en la base de datos a los usuarios a los que el listener escucha a la espera de que escriban sus mensajes. Si el usuario ya existe en la base de datos se actualizará por si éste hubiera cambiado su foto o alguno de sus atributos. Esto se va a realizar de esta manera debido a que los usuarios a los que deberá escuchar el listener serán los usuarios seguidos por una cuenta creada en Twitter específicamente para el proyecto, de modo que si en la cuenta se sigue a un nuevo usuario también deberá aparecer en la base de datos para que el portlet cliente pueda mostrar sus mensajes.

La clase posee dos atributos utilizados para realizar la conexión con la base de datos y como métodos el constructor donde se realiza la conexión, un método para cerrarla y un otro método con el que permitirá introducir nuevos usuarios o actualizar los antiguos se estos ya se encuentran en la base de datos.

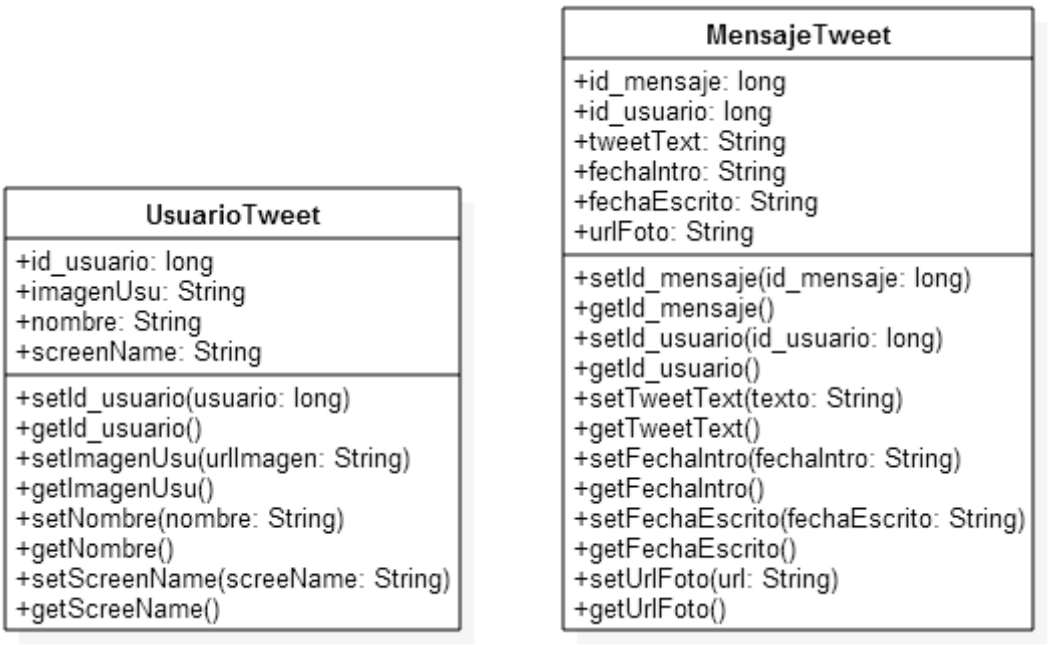

Ilustración 47. Clase UsuarioTweet y MensajeTweet.

Las dos clases de dominio poseen los atributos de un mensaje y de un usuario respectivamente y como métodos los set y get de cada atributo.

Como ya hemos visto anteriormente en el otro componente que estaba diseñado siguiendo la arquitectura de tres capas, la clase que implementa el hook (que en este caso sería la capa de servicio) está relacionada con las clases de de las otras dos capas ya que es la que gestiona todo el procedimiento de introducción de mensajes en la base de datos, por lo tanto tiene que estar conectada con ambas capas.

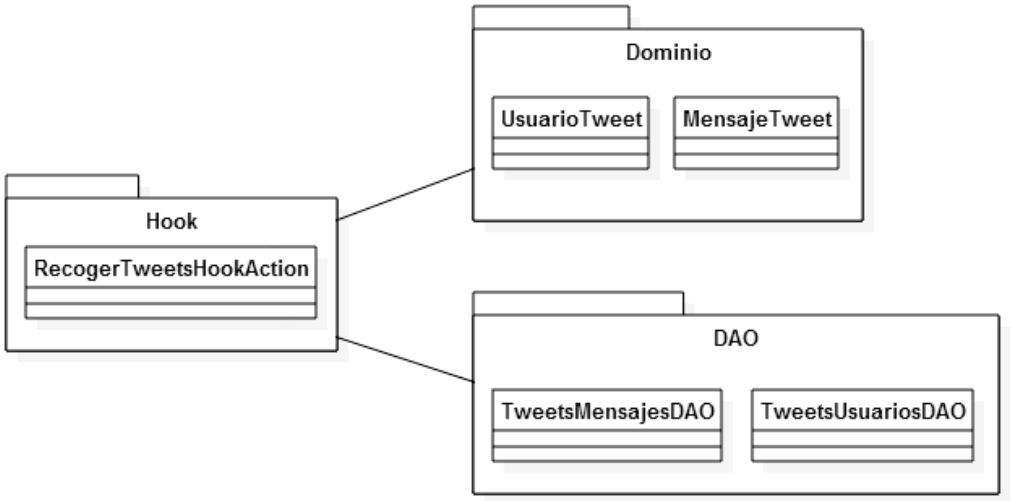

Ilustración 48. Diagrama de clases del hook de Liferay.

Por último se va a mostrar el diagrama dinámico de secuencia para comprender mejor el funcionamiento del hook.

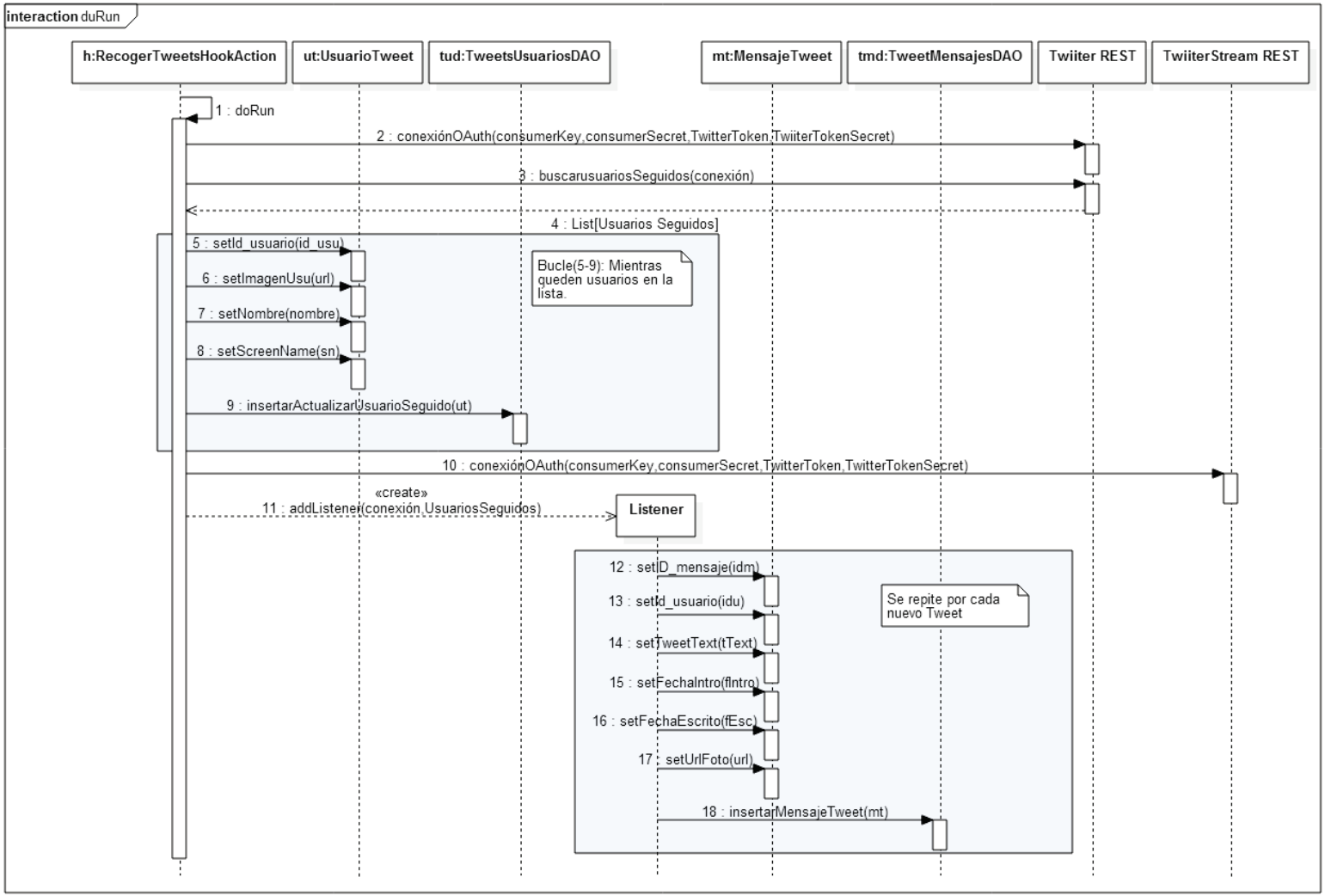

Ilustración 49. Diagrama de secuencia del hook de Liferay.

El hook, después de ser lanzado se conecta a la API REST de Twitter mediante el protocolo de autorización de acceso OAuth. Este protocolo permite a un proveedor dar acceso a un consumidor sin que éste último deba compartir toda su identidad. Está especialmente dirigido parara desarrolladores de consumidores. Para conectarse a Twitter es necesario haber registrado la aplicación a desarrollar y haber recibido 4 claves de acceso para el usuario que registra la aplicación. Este usuario, en el caso concreto del proyecto, será el mismo del que el listener escuchará a sus usuarios seguidos.

Una vez conectado se obtienen los usuarios de los que se pretende escuchar sus mensajes. Mediante un bucle se obtendrán todos sus atributos en cada usuario y se introducirán o actualizarán en la base de datos.

Posteriormente se hace la conexión con la API Stream de Twitter que es la que permite configurar los listener. Una vez conectada y añadido el listener, cada vez que llegue un mensaje que haya que recoger, se obtendrán sus atributos y se introducirá el mensaje en la base de datos

A continuación se va a exponer el diseño de los componentes que estarán a la vista de los usuarios del portal y que serán los contenedores de toda la información gestionada en los anteriores componentes que se acaban de explicar: los portlets.

Todos los portlets tienen la misma estructura de clases por lo que lo que se va a hacer es explicar el diseño estático de un portlet general y a partir de ahí el diseño dinámico de cada uno de ellos

Cuando se va a desarrollar un portlet, éste en principio sólo posee una clase principal que extiende la clase *genericPortlet* que es la que permite utilizar los métodos que hacen del proyecto un portlet y no una aplicación web normal.

Esta clase va a poseer 4 métodos, los 4 métodos básicos de un portlet: el método *processAction()* que es el método que se ejecuta cuando el portal recibe una petición de acción. Una petición de acción se da cuando hay que realizar alguna acción previa antes de que se vuelva a mostrar la información por pantalla, como introducir datos en la base de datos, procesar un formulario, etc. Los otros tres métodos son el *doView, doEdit* y *doHelp*, los tres métodos delegados por el método *render*. Los métodos de render se activan cuando el portal recibe una petición de visualización, es decir cuando se requiere modificar la vista del portlet pero sin la necesidad de realizar ningún cambio previo, como por ejemplo al pinchar en un enlace o ir a otra sección de información. El método *doView* es el que se activa cuando se desea cambiar la información mostrada en el portlet pero dentro del propósito general del mismo, es decir mostrar información para la cual está programado. Los otros dos métodos permiten visualizar información sobre el portlet pero no sobre el propósito del portlet. El método *doHelp* se puede programar para mostrar cualquier información necesaria para entender el funcionamiento del portlet, mientras que el método *doEdit* puede ser programado para permitir modificar aspectos de configuración del portlet. Habrá otros dos métodos pero que no son sobrescritos que son el método de *Init* y el de *destroy* para iniciar y destruir el portlet al desplegarlo y replegarlo

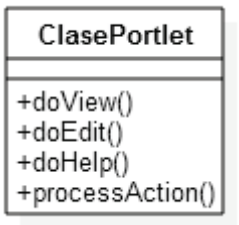

Ilustración 50. Clase principal de un portlet.

Aparte de la clase principal, el resto de los componentes que componen los portlets a desarrollar serán archivos JSP, archivos invocados por los métodos de render que junto con documentos JavaScript, CSS e imágenes, serán los archivos donde se realizará toda la maquetación de la información a mostrar, generando así los fragmentos de código que serán devueltos por el contenedor de portlets para mostrar por pantalla.

A continuación se mostrarán los diagramas de secuencia de cada portlet a desarrollas para explicar el funcionamiento da cada uno de ellos. Cada diagrama mostrará el funcionamiento de cada uno por separado pero hay que tener en cuenta que cuando se

realiza una petición de acción o de visualización sobre uno de los portlets realmente se está haciendo también la fase de render de nuevo sobre el resto de los portlets. Esto ya fue comentado en el apartado Tecnologías cuando se explicaron los portlets.

5.4.5 Portlet de clasificación de la LFP.

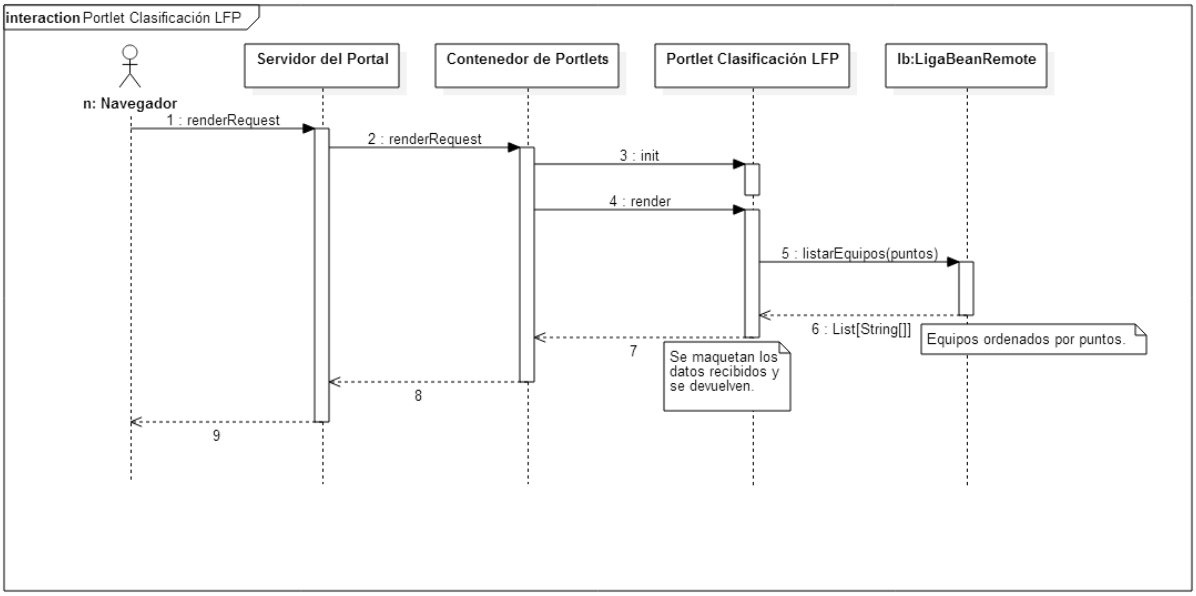

Ilustración 51. Diagrama de secuencia portlet Clasificación LFP.

El portlet se despliega y se activa el método de visualización (el método *doView* delegado por el método *render*). Este método invoca el JSP principal del portlet que hace la llamada al módulo EJB cliente del servicio web de la liga. Devolverá el listado con los equipos ordenados por puntos. En el JSP se maqueta esa información en una tabla ordenable por otros campos y se devolverá el fragmento de código generado al navegador para que se muestre por pantalla.

5.4.6 Portlet de estadísticas de jugadores de la LFP.

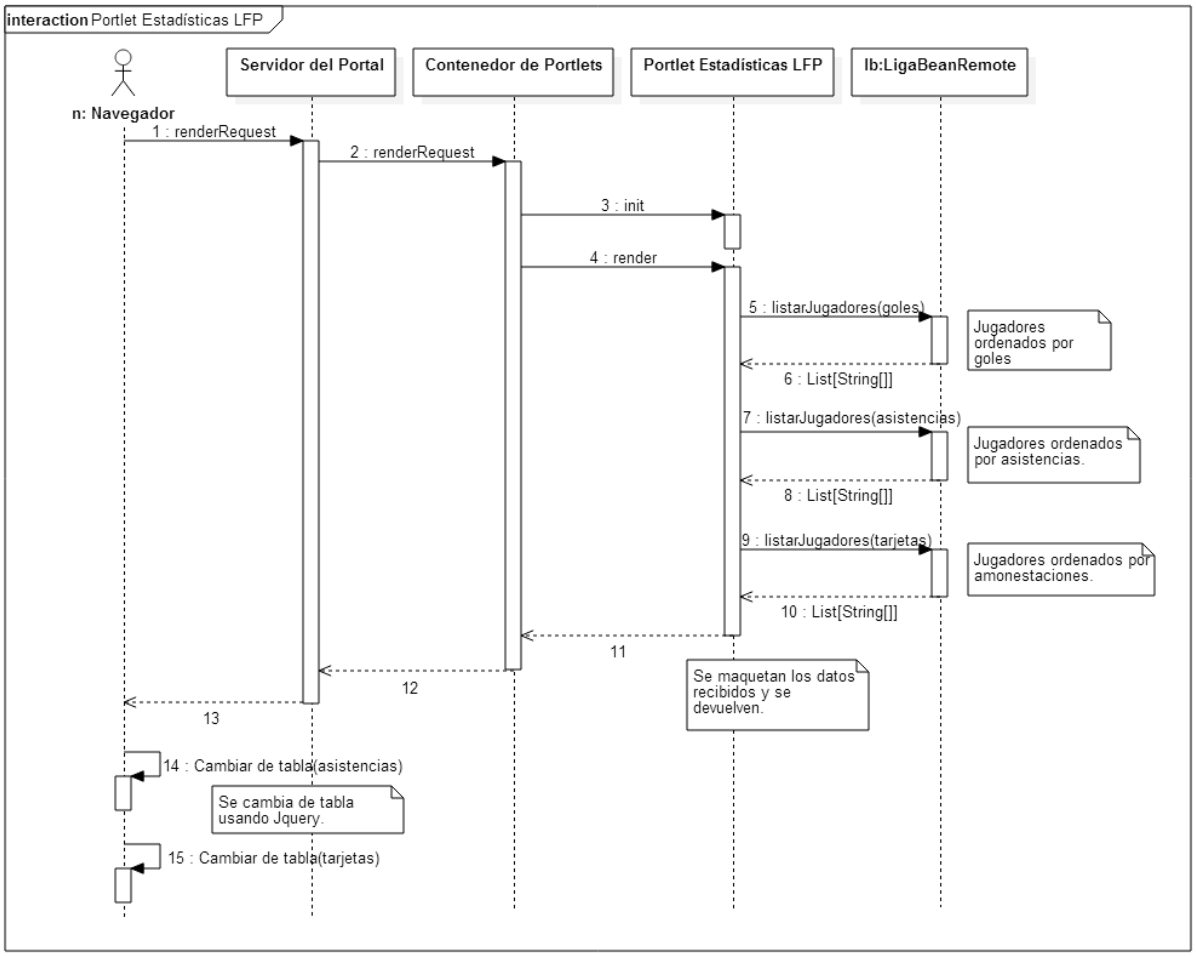

Ilustración 52. Diagrama de secuencia portlet estadísticas LFP.

El portlet se despliega y se activa el método de visualización (el método doView delegado por el método render). Este método invoca el JSP principal del portlet que hace tres llamadas al módulo EJB cliente del servicio web de la liga. Obtendrá un listado de los jugadores ordenado por goles otro por asistencias y otro por tarjetas. En el JSP se maqueta esa información en tres tablas que posteriormente con eventos de JQuery se irán ocultando y mostrando una u otra según la elección del usuario. Como esto se realiza con código JavaScript no es necesario volver a hacer ninguna petición de visualización al servidor.

5.4.7 Portlet de plantillas de la LFP.

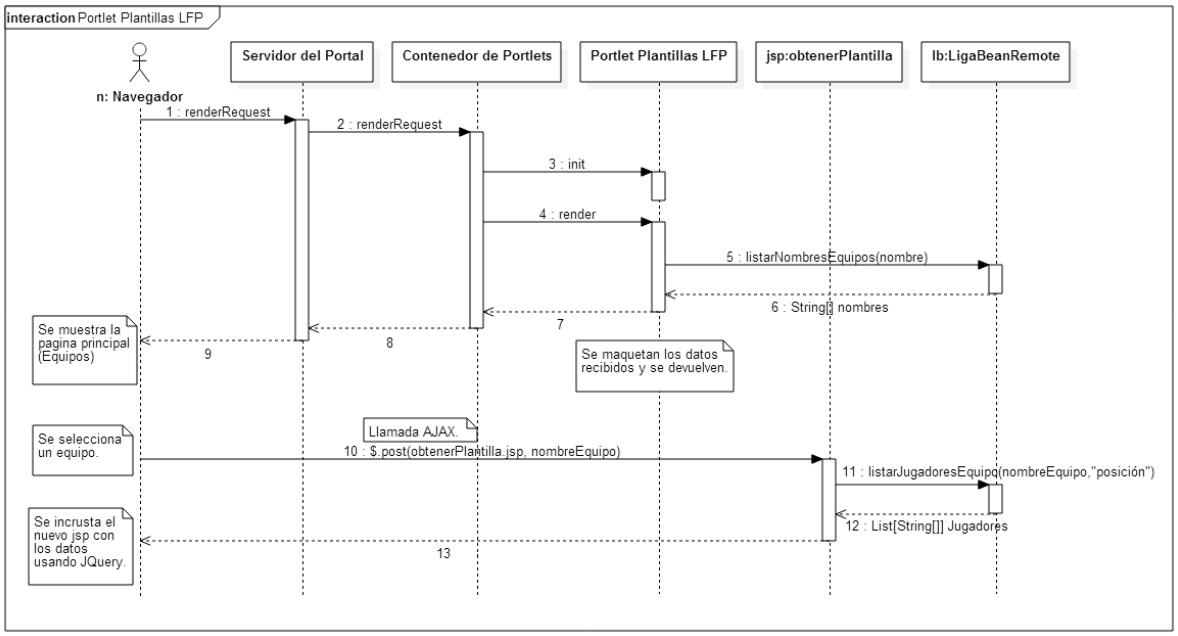

Ilustración 53. Diagrama de secuencia portlet plantillas LFP.

El portlet se despliega y se activa el método de visualización (el método *doView* delegado por el método *render*). Este método invoca el JSP principal del portlet que hace la llamada al módulo EJB cliente del servicio web de la liga que devolverá el listado con los nombres de los equipos. En el JSP se maqueta esa información para que se vea una matriz con los escudos de todos los equipos y se devuelve al navegador para que se muestre por pantalla.

Una vez se muestra la matriz en pantalla, si el usuario pincha sobre una de los escudos se realizará una petición Post AJAX de JQuery para invocar un JSP que se encargue de pedir la información de la plantilla seleccionada al EJB cliente del servicio web de la liga. Este JSP maquetará la información de la platilla y devolverá la información directamente al navegador sin realizar de nuevo ninguna petición de visualización que provocaría actualizar toda la página del portal (ya que activaría el método de visualización de todos los portlets).

5.4.8 Portlet lector de Twitter.

Este portlet además de la clase general que poseen todos los portlets tiene dos clases DAO ya que el portlet se va a conectar a la base de datos para obtener los mensajes y los usuarios que escribieron estos. De esta forma se separa la parte de la conexión a la base de datos del funcionamiento del portlet.

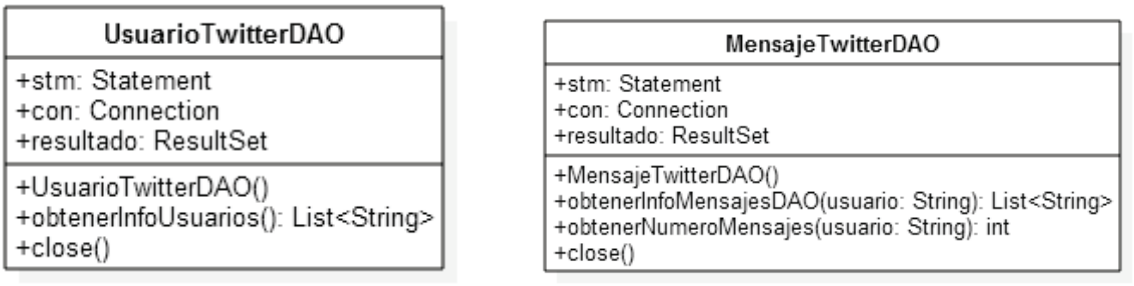

Ilustración 54. Clases UsuarioTwitterDAO y MnesajeTwitterDAO.

Ambas clases, además de los atributos para conectar y los métodos constructor y *close* donde se realiza la conexión y se cierra la misma, poseerán los métodos para obtener la información: el método *obtenerInfoUsuarios* para obtener el screenName y el id del usuario en la clase UsuarioTwitterDAO y los métodos *obtenerInfoMensajes* y *obtenerNumeroMensajes* para obtener la información de los mensajes y para contar el número de mensajes respectivamente en la clase MensajeTwitterDAO.

Después de explicar esta particularidad de este portlet en comparación con el resto descritos anteriormente se procederá, ahora sí a describir su diagrama de secuencia.

En el diagrama que se muestra en la página siguiente, el portlet se despliega y se activa el método de visualización (el método *doView* delegado por el método *render*). Este método invoca el JSP principal del portlet que hará una petición Post AJAX de JQuery para invocar un nuevo JSP que se encargará de maquetar el selector de usuarios con los usuarios obtenidos de la base de datos y devolverlo al primer JSP. El fragmento de código del selector se introducirá en el árbol DOM del primer JSP. Posteriormente se hará otra llamada Post AJAX que invocará un JSP nuevo para obtener los mensajes de la base de datos, maquetarlos y devolverlos al primer JSP. El código del panel de mensajes generado se coloca en el árbol DOM. Una vez todo junto se devuelve el fragmento de código para visualizarlo en pantalla.

Mediante un temporizador se harán invocaciones periódicas post de AJAX para recargar el panel de mensajes sin realizar ninguna petición de visualización y por lo tanto, sin actualizar todo el portal. Lo mismo ocurrirá cuando se cambie de usuario o de página en el panel.

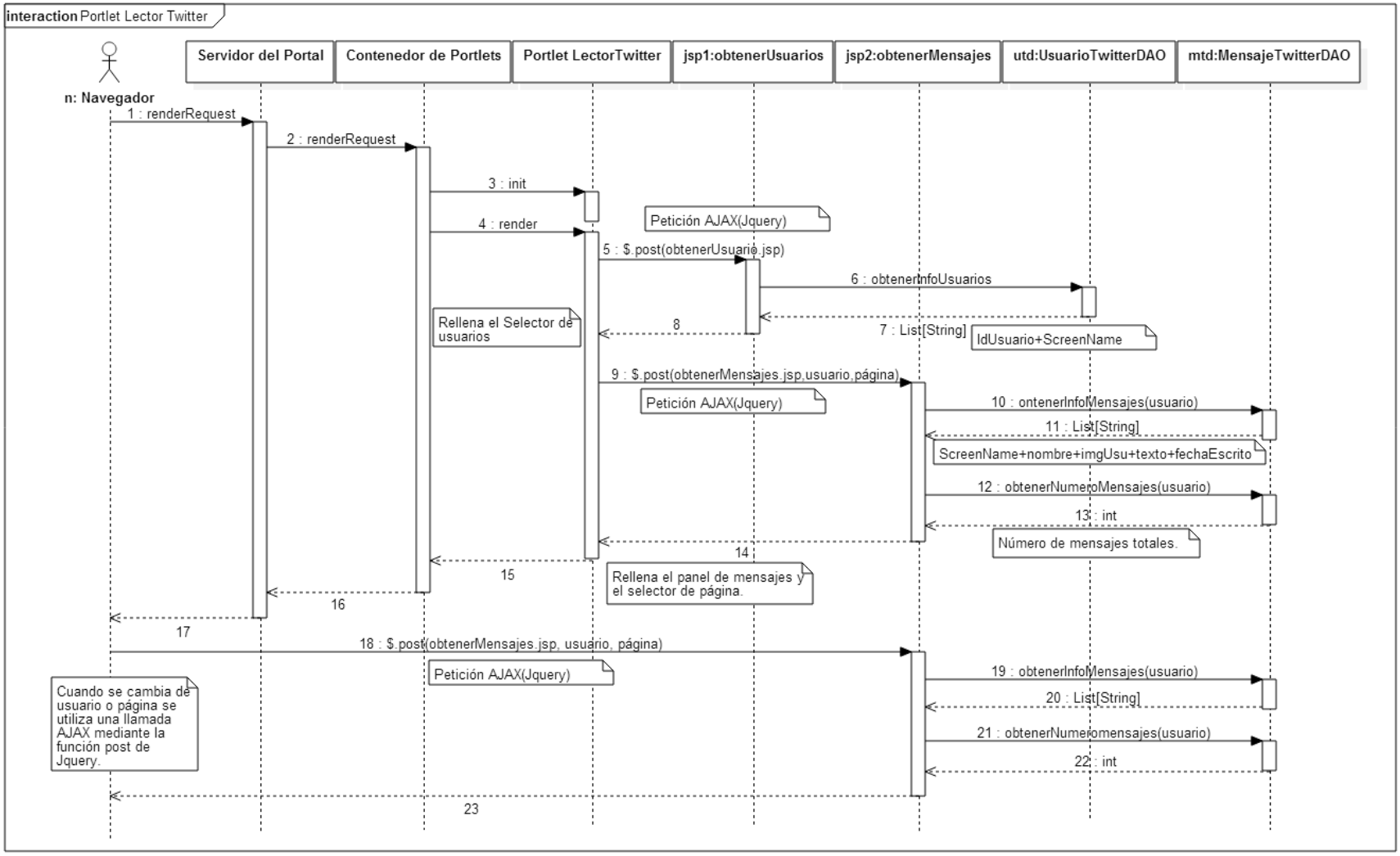

Ilustración 55. Diagrama de secuencia portlet lector de Twitter.

5.4.9 Slideshow lector de canales RSS.

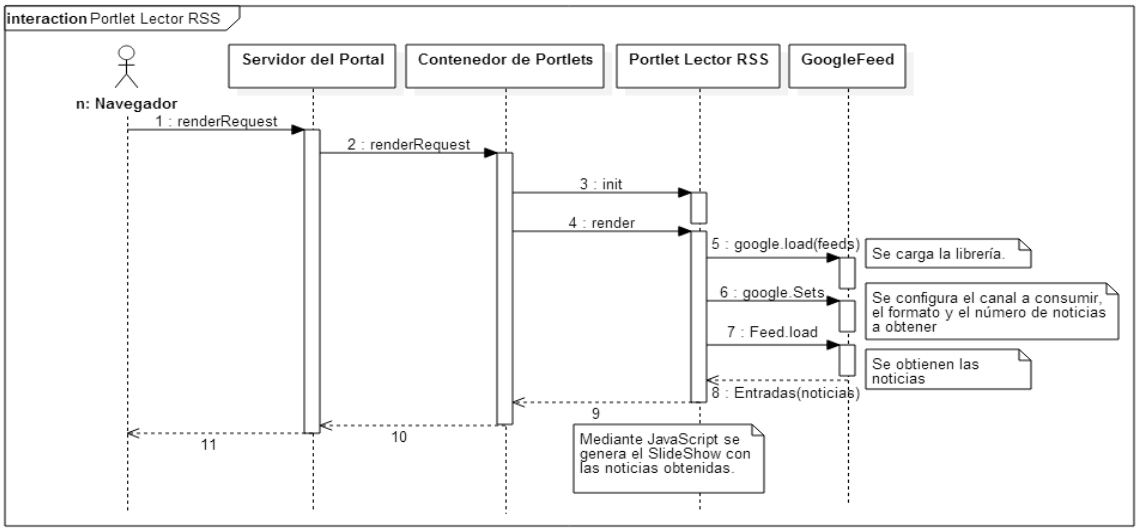

Ilustración 56. Diagrama de secuencia portlet lector RSS.

El portlet se despliega y se activa el método de visualización (el método *doView* delegado por el método *render*). Este método invoca el JSP principal del portlet. El JSP se va a encargar de todo el funcionamiento del portlet utilizando la librería Feed de Google. Esta es una librería para lenguaje JavaScript que permite manejar y gestionar toda la información de un canal RSS simplemente usando JavaScript. Para utilizarla, el JSP principal del portlet realizará en primer lugar la carga de la librería, después realizará las configuraciones necesarias (indicando que canal RSS consumir, el formato de devolución de los datos del canal y el número de ítems) y por último iniciará la función que gestionará los datos devueltos.

En esta función se recorrerán los elementos devueltos obteniendo la información deseada de cada uno y realizando la maquetación en forma de slideshow. Para que los diferentes elementos que se mostrarán aparezcan en forma de slideshow, se introducirán todas las entradas en el panel principal y se utilizarán los diferentes eventos de JQuery para ocultar y mostrar elementos dejando únicamente visible el elemento activo. También se utilizara la librería JQuery User Interface para resaltar el elemento activo. Para provocar el cambio de elemento a mostrar se utilizará un temporizador que ocultará el elemento activo y mostrará el siguiente elemento cada 8 segundos. Habrá otro temporizador para recargar el canal y mostrar siempre el panel actualizado. Éste estará configurado para actualizar cada 30 minutos. Ambos temporizadores comprobarán el tamaño que ocupará el portlet en pantalla para mostrar un número de elementos y una disposición de los mismos diferente según el tamaño. También se configurarán eventos para el momento en que el usuario pase en ratón por encima de las noticias mostradas, parando el temporizador de giro momentáneamente y haciendo foco sobre el elemento seleccionado. El portlet también mostrará un selector para que el usuario pueda cambiar el canal a mostrar. Este cambio de canal reinicia todo el proceso con el nuevo canal. Como estará todo programado mediante JavaScript ni el temporizador de giro ni el de actualización del canal, ni tampoco cuando el usuario elija otro canal diferente provocarán que el portal se actualice entero.

5.4.10 Goleadores Mundial de fútbol de Brasil 2014.

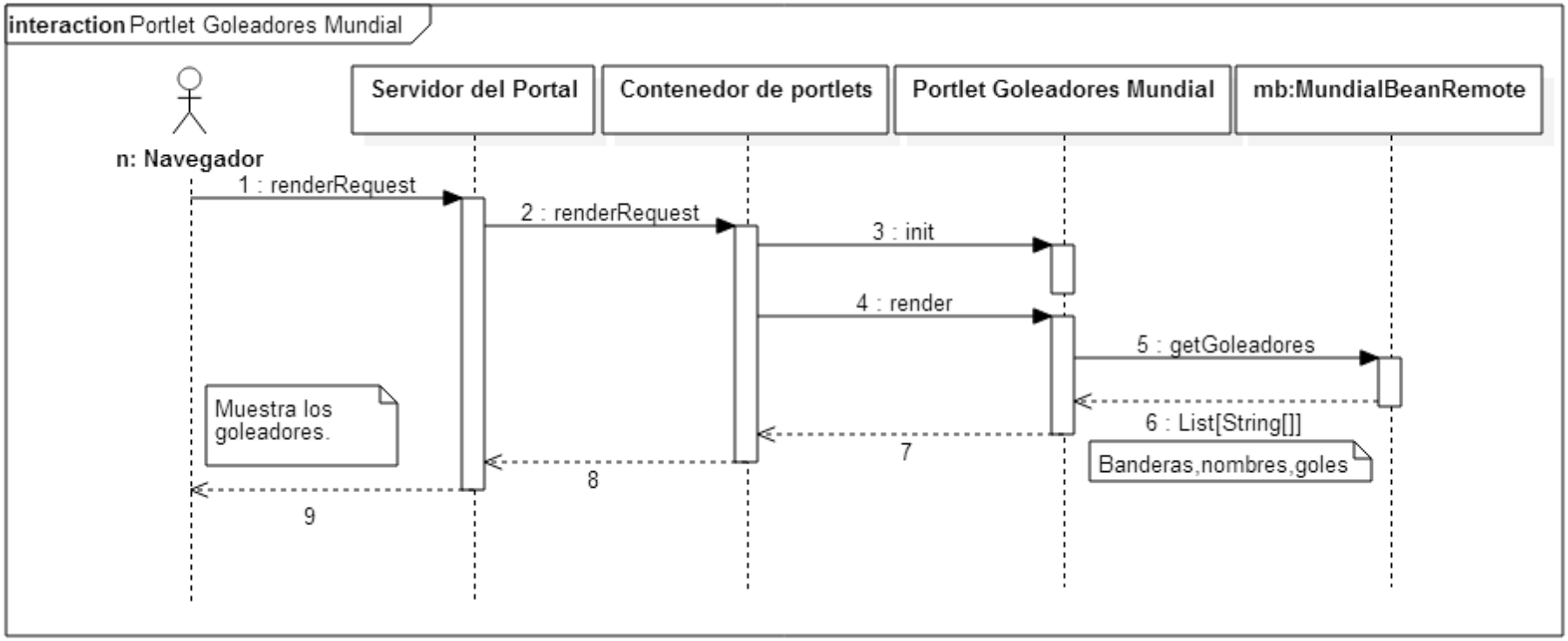

Ilustración 57. Diagrama de secuencia portlet goleadores mundial 2014.

El portlet se despliega y se activa el método de visualización (el método doView delegado por el método render). Este método invoca el JSP principal del portlet que hace la llamada al módulo EJB cliente del servicio web del Mundial de Brasil 2014, concretamente al método que devolverá la lista de goleadores con la bandera de sus países, los nombres y el número de goles. El JSP lo maqueta y devuelve el fragmento de código para que se muestre por pantalla.

5.4.11 Información general Mundial de fútbol de Brasil 2014.

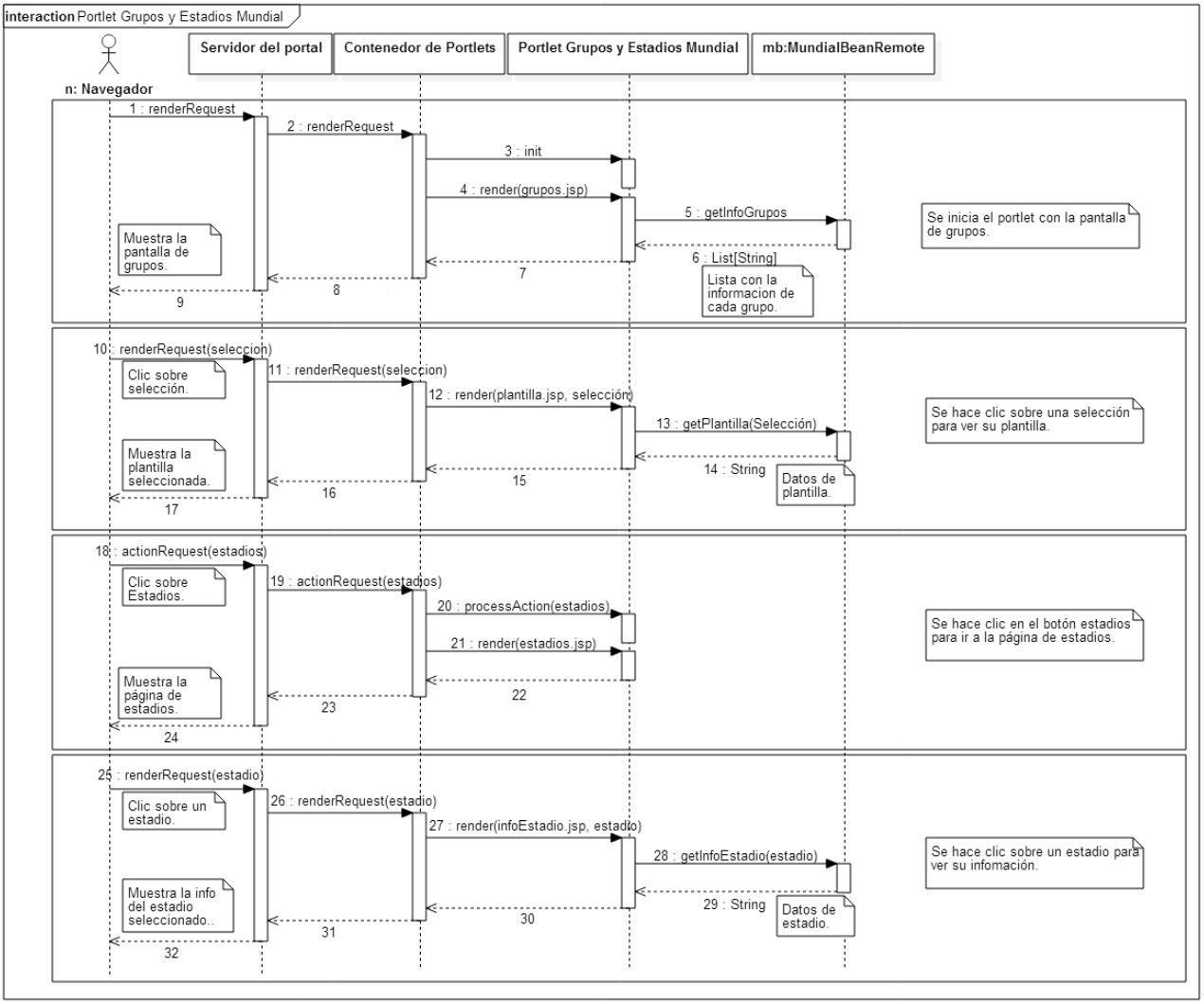

Ilustración 58. Diagrama de secuencia portlet información general mundial 2014.

El portlet se despliega y se activa el método de visualización (el método *doView* delegado por el método *render*). Este método invoca el JSP principal del portlet que hace la llamada al módulo EJB cliente del servicio web del Mundial de Brasil 2014 para obtener la información de todos los grupos. El JSP lo maqueta y devuelve el fragmento de código para que se visualice en el navegador.

Posteriormente un usuario hace clic sobre una selección para ver su plantilla por lo que el servidor del portal recibe la petición indicando al contenedor de portlets que active el método de visualización adecuado (el método *doView*) sobre el portlet. Este método invoca el JSP de las plantillas. El JSP se encarga de llamar al método *getPlantilla* del EJB que obtiene todos los miembros de la plantilla pasada como parámetro. Después maqueta la información mostrando los jugadores separados por posición y devuelve el fragmento de código para que se muestre en el navegador. A la vez de activar el método de visualización sobre este portlet el contenedor de portlets también habrá activado el método de render de todos los portlets que estén desplegados en el portal.

A continuación el cliente hace clic en el botón de estadios. El servidor le pasará la petición al contenedor de portlets para que este active el método de procesamiento de acciones (*processAction*) que prepara el terreno para que el posterior método de visualización que se lanzará obligatoriamente después de haber finalizado el *processAction* invoque el JSP de los estadios. Al mismo tiempo se ha activado el método de visualización de todos los portlets desplegados en el portal. Una vez han finalizado los métodos de visualización de todos los portlets se devuelven los fragmentos de código y se muestra en el navegador.

Por último el cliente pincha sobre uno de los estadios para acceder a su información El servidor del portal avisa al contenedor de portlets de que active el método *doView* del portlet que invocará el JSP de información de estadio. El JSP llamará al método del EJB que le devuelve todos los datos del estadio pasado como parámetro, maqueta la información incluyendo el mapa de Google maps de la zona del estadio utilizando la API maps de Google. Después devuelve el fragmento de código, que llega al contenedor junto con el fragmento de cada portlet desplegado que han realizado también sus métodos de visualización, para finalmente enviarlos al navegador y visualizarlos en la pantalla.

Este portlet está desarrollado sin utilizar tecnología AJAX, por lo que como se ha comentado al final de cada párrafo se realiza el renderizado de todos los portlets después de cada petición de acción y visualización. Esto provoca que se recargue toda la página del portal cada vez que se hace una acción sobre este portlet. Es un muy buen ejemplo para demostrar la importancia de utilizar AJAX en el desarrollo de portlets.

5.5 Diseño de la Base de datos

Cuando se despliega por primera vez el módulo Liferay sobre el servidor de aplicaciones es necesario tener instalado el servidor de bases de datos que se desee utilizar ya que es en este primer despliegue cuando Liferay crea sus tablas con toda la información necesaria para que el portal funcione. Este esquema creado por Liferay no va a ser comentado ya que ha sido totalmente creado por Liferay. Lo que si se va a mostrar en este apartado es el diseño del pequeño esquema creado para el funcionamiento de los componentes que forman parte del proyecto a desarrollar.

Como se ha comentado durante la descripción del diseño del sistema, existen varios componentes que, tanto para introducir como para obtener datos, deben acceder a la base de datos. Estos eran por un lado el hook que introducía mensajes y los usuarios de Twitter que los escribían y el portlet lector de Twitter que los obtenía posteriormente y por otro lado el servicio web que accedía a equipos y jugadores de la LFP para devolver la información a los clientes. Teniendo en cuenta la información que se va a tratar, se han diseñado los siguientes diagramas del esquema de la base de datos que utilizan los componentes desarrollados.

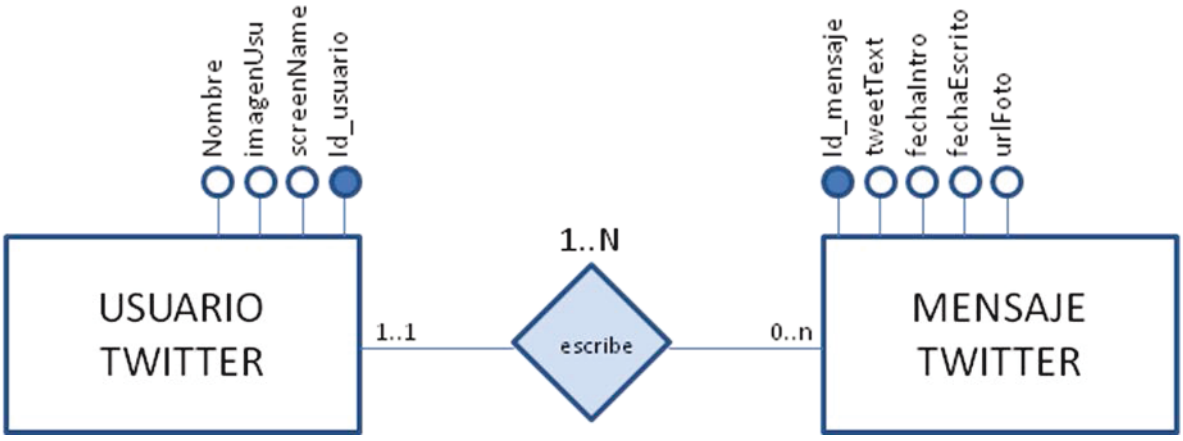

Ilustración 59. Diagrama E/R tablas Twitter.

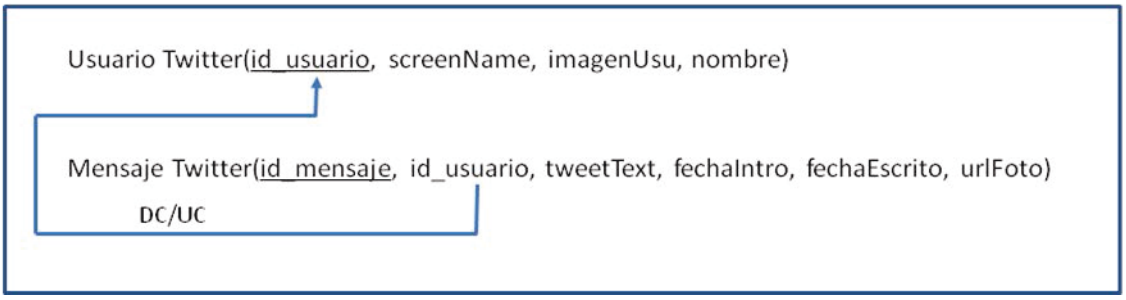

Ilustración 60. Grafo relacional tablas Twitter.

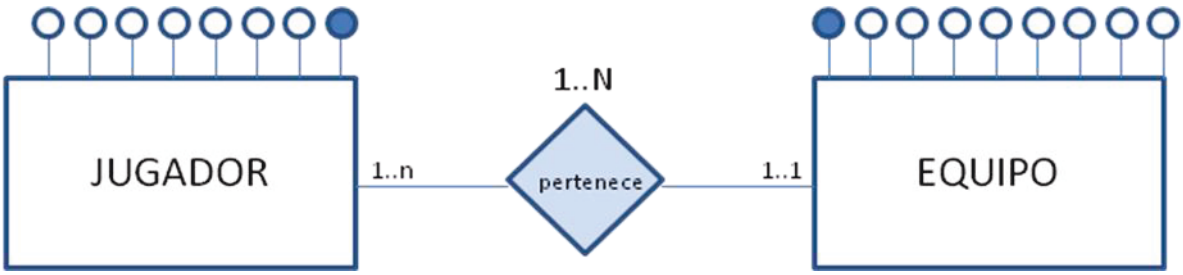

Ilustración 61. Diagrama E/R tablas Servicio web LFP.

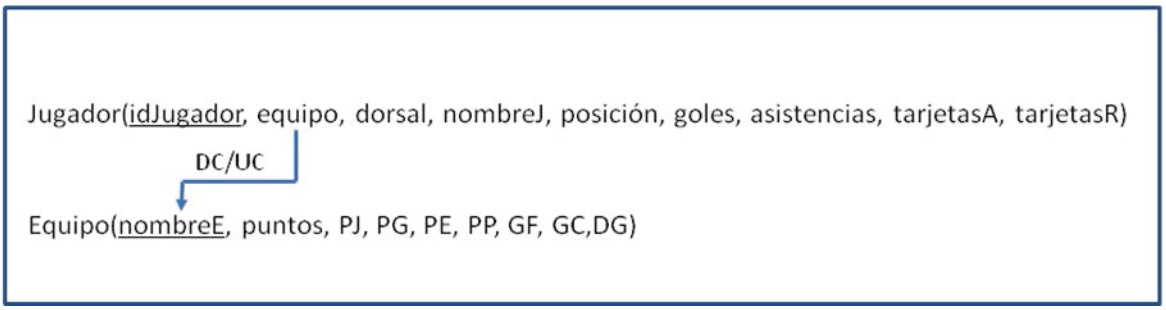

Ilustración 62. Grafo relacional tablas Servicio web LFP.

5.6 Conclusión

Terminada la fase de diseño donde se ha descrito la arquitectura física y lógica del sistema así como el diseño individual de cada uno de los componentes que lo conforman, se puede comenzar con la fase de desarrollo. Esta fase no se va a exponer en la memoria por lo que en el siguiente capítulo se expone la fase de pruebas, donde se incluyen las pruebas realizadas sobre el proyecto una vez terminada la fase de desarrollo.

Capítulo 6

Pruebas

6.1 Introducción

Con el objetivo de asegurar que el software creado cumple con las especificaciones requeridas obtenidas en la fase de análisis, se han diseñado y realizado una serie de pruebas que a su vez sirven para detectar y eliminar errores que no se hayan detectado previamente.

En este capítulo se recogen las principales pruebas funcionales que se han realizado sobre los diferentes portlets desarrollados. Todas ellas se han evaluado de forma manual, es decir, las ejecuta un tester como si fuese un usuario normal, siguiendo el plan de pruebas diseñado en la fase de análisis de los requisitos con el objetivo de garantizar que la aplicación hace lo que debe.

Estas pruebas se han realizado tras la conclusión del desarrollo de cada portlet, y se han repetido hasta que todas ellas tenían un resultado satisfactorio. De esta forma no se avanza si alguna de las partes no funciona como se espera.

En el caso de que alguna prueba no cumpla con la salida prevista se marca como pendiente y después de la corrección del código se vuelve a realizar hasta que todas tengan el estado concluido. Para que no quede demasiado engorroso, en este documento simplemente se exponen las pruebas una vez que todas ellas fueron satisfactorias, de esta forma no se repiten las tablas.

6.2 Plan de pruebas

Para cada caso de uso y sus requisitos relacionados obtenidos en la fase de análisis, se recoge una tabla con todas las pruebas realizadas para probar su correcto funcionamiento. Cada una de estas tablas recoge una serie de información que identifica y describe cada una de las pruebas realizadas. A continuación se muestra una descripción de la información que ofrece cada tabla.

Caso de uso a probar		*Identificador y nombre del caso de uso a probar.*			
ID Caso de prueba	**Descripción del caso de prueba**	**Pre requisitos**	**Resultado esperado**	**Resultado obtenido**	**Estado**
Identificador de prueba	*Define lo que se pretende verificar*	*Condiciones previas para realizar la prueba*	*Previsión de la salida*	*Salida real*	*Concluido/ Pendiente*

Tabla 3. Descripción general de tabla de pruebas.

Cómo los requisitos no funcionales no se recogen en estas tablas, una vez terminada la fase de desarrollo se hizo una revisión uno a uno de estos requisitos para verificar que se hubieran satisfecho.

6.3 Pruebas

Caso de uso a probar		CU-003 Visualizar tweets			
ID Caso de prueba	Descripción del caso de prueba	Pre requisitos	Resultado esperado	Resultado obtenido	Estado
CP-001	Verificar que la información se actualiza en tiempo real.	Usuario en página principal. Portlet lector Twitter desplegado.	Aparecen nuevos tweets	Aparecen nuevos tweets	Concluido.
CP-002	Verificar que la maquetación del tablón de tweets se optimiza al tamaño del portlet.	Usuario administrador o miembro cambia de lugar el portlet.	La maquetación se optimiza al ancho del portlet.	La maquetación se optimiza al ancho del portlet.	Concluido.
CP-003	Verificar que los enlaces que aparezcan en los tweets se transformen en hipervínculos, con su estilo característico.	Existe un tweet con un enlace en el texto.	El enlace se transforma en un hipervínculo.	El enlace se transforma en un hipervínculo.	Concluido.
CP-004	Verificar que los enlaces de los tweets se abren en una pestaña diferente.	Existe un tweet con un enlace en el texto.	El enlace se abre en otra pestaña.	El enlace se abre en otra pestaña.	Concluido.

Tabla 4. Pruebas CU-003 Visualizar tweets.

Caso de uso a probar		CU-004 Cambiar página tablón			
ID Caso de prueba	Descripción del caso de prueba	Pre requisitos	Resultado esperado	Resultado obtenido	Estado
CP-005	Verificar que un usuario puede cambiar la página del tablón de tweets.	Portlet lector Twitter desplegado. Usuario cambia la página.	Se muestran los tweets de la página correspondiente.	Se muestran los tweets de la página correspondiente.	Concluido.
CP-006	Verificar que se puede cambiar de página del tablón de tweets a la vez que se filtran los tweets por usuario.	Se muestran los tweets de un usuario concreto. Usuario cambia la página.	Se muestran los tweets del usuario seleccionado y de la página correspondiente.	Se muestran los tweets del usuario seleccionado y de la página correspondiente.	Concluido.
CP-007	Verificar que la información se actualiza en tiempo real teniendo en cuenta los filtros activos.	Se han filtrado los tweets por usuario y página.	Aparecen nuevos tweets manteniendo las opciones de filtrado.	Aparecen nuevos tweets manteniendo las opciones de filtrado.	Concluido.

Tabla 5. Pruebas CU-004 Cambiar página tablón.

Caso de uso a probar		CU-005 Filtrar tweets por usuario			
ID Caso de prueba	Descripción del caso de prueba	Pre requisitos	Resultado esperado	Resultado obtenido	Estado
CP-008	Verificar que un usuario puede filtrar los tweets por un usuario de Twitter concreto.	Portlet lector Twitter desplegado. El usuario cambia de usuario de Twitter.	Se muestran los tweets del usuario correspondiente.	Se muestran los tweets del usuario correspondiente.	Concluido.
CP-009	Verificar que la información se actualiza en tiempo real con el filtro de usuario activo.	Visualizando tweets de un usuario de Twitter concreto.	Aparecen nuevos tweets y se mantiene el usuario.	Aparecen nuevos tweets y se mantiene el usuario.	Concluido.
CP-010	Verificar que la página del tablón se reinicia cuando se cambia de usuario.	Tablón en página antigua. El usuario cambia de usuario de Twitter.	Página del tablón se reinicia en 1, mostrando los mensajes más modernos.	Página del tablón se reinicia en 1, mostrando los mensajes más modernos.	Concluido.

Tabla 6. Pruebas CU-005 Filtrar tweets por usuario

Caso de uso a probar		CU-006 Ver tweet con foto			
ID Caso de prueba	Descripción del caso de prueba	Pre requisitos	Resultado esperado	Resultado obtenido	Estado
CP-011	Verificar que un usuario puede expandir un tweet con foto.	Usuario hace clic sobre la foto.	Se abre el tweet en una ventana modal.	Se abre el tweet en una ventana modal.	Concluido.

Tabla 7. Pruebas CU-006 Ver tweet con foto.

Caso de uso a probar		CU-007 Visualizar feeds RSS			
ID Caso de prueba	Descripción del caso de prueba	Pre requisitos	Resultado esperado	Resultado obtenido	Estado
CP-012	Verificar que el slideshow cambia el foco de la noticia cada 7sg.	Usuario en página principal.	Se cambia el foco de una noticia activa a la siguiente.	Se cambia el foco de una noticia activa a la siguiente.	Concluido.
CP-013	Verificar que el slideshow se reinicia cuando llega a la última noticia.	Usuario en página principal.	El foco vuelve a la primera noticia.	El foco vuelve a la primera noticia.	Concluido.
CP-014	Verificar que la maquetación del slideshow de feed RSS se optimiza al tamaño del portlet.	Usuario administrador o miembro cambia de lugar el portlet.	La maquetación se optimiza al ancho del portlet.	La maquetación se optimiza al ancho del portlet.	Concluido.
CP-015	Verificar el slideshow se actualiza automáticamente cada 30minutos para mostrar las últimas noticias.	Usuario en página principal. Portlet lector RSS activo.	Aparecen nuevas noticias.	Aparecen nuevas noticias.	Concluido.

Tabla 8. Pruebas CU-007 Visualizar feeds RSS.

Caso de uso a probar		CU-008 Hacer foco			
ID Caso de prueba	Descripción del caso de prueba	Pre requisitos	Resultado esperado	Resultado obtenido	Estado
CP-016	Verificar que cuando se seleccione un titular del slideshow con el ratón se muestre su información.	Usuario selecciona una de las noticias del slideshow.	Se muestra la información ampliada.	Se muestra la información ampliada.	Concluido.
CP-017	Verificar que cuando se seleccione un titular del slideshow la rotación se detiene.	Usuario selecciona una de las noticias del slideshow.	La rotación se detiene	La rotación se detiene	Concluido.
CP-018	Verificar que cuando se saca el ratón de los titulares la rotación comienza de nuevo.	Usuario saca el ratón de los titulares.	La rotación comienza desde el último titular con foco.	La rotación comienza desde el último titular con foco.	Concluido.

Tabla 9. Pruebas CU-008 Hacer foco.

Caso de uso a probar		CU-009 Acceder noticia completa			
ID Caso de prueba	Descripción del caso de prueba	Pre requisitos	Resultado esperado	Resultado obtenido	Estado
CP-019	Verificar que cuando se hace clic sobre un titular del slideshow se abre una nueva pestaña con la fuente original.	Usuario hace clic en un titular.	Nueva pestaña con la fuente original.	Nueva pestaña con la fuente original.	Concluido.

Tabla 10. Pruebas CU-009 Acceder noticia completa.

Caso de uso a probar		CU-010 Seleccionar canal RSS			
ID Caso de prueba	Descripción del caso de prueba	Pre requisitos	Resultado esperado	Resultado obtenido	Estado
CP-020	Verificar que un usuario puede cambiar el feed RSS para ver las noticias de otro medio.	Usuario cambia el feed RSS eligiendo una de las opciones que se ofrecen.	El slideshow mostrará la información del medio seleccionado.	El slideshow mostrará la información del medio seleccionado.	Concluido.
CP-021	Verificar que el portlet se actualiza automáticamente aunque se cambie el feed a mostrar.	Usuario cambia el feed RSS.	Aparecen nuevas noticias del medio seleccionado.	Aparecen nuevas noticias del medio seleccionado.	Concluido.

Tabla 11. Pruebas CU-010 Seleccionar canal RSS.

Caso de uso a probar		CU-012 Visualizar grupos			
ID Caso de prueba	Descripción del caso de prueba	Pre requisitos	Resultado esperado	Resultado obtenido	Estado
CP-022	Verificar que al cargar el portlet se muestra la página principal con los grupos de selecciones del mundial.	-	Se muestra la página de grupos.	Se muestra la página de grupos.	Concluido.
CP-023	Verificar que la maquetación de la tabla de grupos se optimiza al tamaño del portlet.	Usuario administrador o miembro cambia de lugar el portlet.	La maquetación se optimiza al ancho del portlet.	La maquetación se optimiza al ancho del portlet.	Concluido.
CP-024	Verificar que se muestra el camino de navegación que indicará en que página se encuentra el usuario.	-	Camino de navegación visible.	Camino de navegación visible.	Concluido.
CP-025	Verificar que el camino de navegación permite moverse a otra página.	Se muestran las diferentes páginas como enlaces. El usuario hace clic en una de las páginas.	Se carga la página seleccionada.	Se carga la página seleccionada.	Concluido.

Tabla 12. Pruebas CU-012 Visualizar grupos.

Caso de uso a probar		CU-013 Visualizar plantillas			
ID Caso de prueba	Descripción del caso de prueba	Pre requisitos	Resultado esperado	Resultado obtenido	Estado
CP-026	Verificar que cuando el usuario hace clic sobre uno de los equipos se muestra la plantilla de ese equipo.	Usuario hace clic sobre una selección.	Nueva página con la plantilla del equipo seleccionado	Nueva página con la plantilla del equipo seleccionado	Concluido.
CP-027	Verificar que el usuario puede volver a la página principal usando el botón volver.	Usuario hace clic sobre el botón volver.	Se muestra la página de grupos.	Se muestra la página de grupos.	Concluido.
CP-028	Verificar que se muestra el camino de navegación en la página de plantillas y que puede usarse.	Usuario hace clic sobre una de las páginas.	Se muestra la página seleccionada.	Se muestra la página seleccionada.	Concluido.

Tabla 13. Pruebas CU-013 Visualizar plantillas.

Caso de uso a probar		CU-014 Visualizar lista estadios			
ID Caso de prueba	Descripción del caso de prueba	Pre requisitos	Resultado esperado	Resultado obtenido	Estado
CP-029	Verificar que el enlace a la página de estadios funciona, cargando la página con la información de los estadios.	Usuario hace clic sobre el botón de estadios.	Se muestra la página de estadios.	Se muestra la página de estadios.	Concluido.

Tabla 14. Pruebas CU-014 Visualizar lista estadios.

Caso de uso a probar		CU-015 Visualizar info estadios			
ID Caso de prueba	Descripción del caso de prueba	Pre requisitos	Resultado esperado	Resultado obtenido	Estado
CP-030	Verificar que las imágenes de los estadios funcionan como enlaces mostrando la información del estadio correspondiente.	Página de estadios cargada. El usuario hace clic sobre unos de los estadios.	Se muestra la página con la información del estadio seleccionado	Se muestra la página con la información del estadio seleccionado	Concluido.
CP-031	Verificar que el botón de volver regresa a la página de selección de estadios.	El usuario hace clic sobre el botón volver.	Se muestra la página de estadios.	Se muestra la página de estadios.	Concluido.
CP-032	Verificar que el enlace de Google Maps se convierte en un mapa incrustado en la maquetación.	El usuario selecciona un estadio.	Se muestra el mapa con la localización del estadio.	Se muestra el mapa con la localización del estadio.	Concluido.

Tabla 15. Pruebas CU-015 Visualizar info estadios.

Caso de uso a probar		CU-018 Visualizar clasificación			
ID Caso de prueba	Descripción del caso de prueba	Pre requisitos	Resultado esperado	Resultado obtenido	Estado
CP-033	Verificar que la información de la tabla de la clasificación se puede ordenar según sus columnas.	El usuario hace clic sobre cualquiera de las columnas ordenables.	Los datos de la tabla se ordenan según la columna seleccionada.	Los datos de la tabla se ordenan según la columna seleccionada.	Concluido.

Tabla 16. Pruebas CU-018 Visualizar clasificación.

Caso de uso a probar		CU-019 Visualizar estadísticas			
ID Caso de prueba	Descripción del caso de prueba	Pre requisitos	Resultado esperado	Resultado obtenido	Estado
CP-034	Verificar que un usuario puede cambiar entre las diferentes tablas de estadísticas(goles, asistencias, amonestaciones)	El usuario hace clic sobre cualquiera de los botones para cambiar de tabla.	Se muestra la tabla de estadísticas correspondiente.	Se muestra la tabla de estadísticas correspondiente.	Concluido.
CP-035	Verificar que la información de las tablas de estadísticas se puede ordenar según sus columnas.	El usuario hace clic sobre cualquiera de las columnas ordenables.	Los datos de la tabla se ordenan según la columna seleccionada.	Los datos de la tabla se ordenan según la columna seleccionada.	Concluido.

Tabla 17. Pruebas CU-019 Visualizar estadísticas.

Caso de uso a probar		CU-020 Visualizar plantillas liga			
ID Caso de prueba	Descripción del caso de prueba	Pre requisitos	Resultado esperado	Resultado obtenido	Estado
CP-036	Verificar que las imágenes de los escudos de los equipos funcionan como enlaces mostrando la información de la plantilla correspondiente.	Página de equipos cargada. El usuario hace clic sobre unos de los escudos.	Se muestra la página con la información de la plantilla seleccionada.	Se muestra la página con la información de la plantilla seleccionada.	Concluido.
CP-037	Verificar que la información de las tablas de plantillas se puede ordenar según sus columnas.	El usuario hace clic sobre cualquiera de las columnas ordenables.	Los datos de la tabla se ordenan según la columna seleccionada.	Los datos de la tabla se ordenan según la columna seleccionada.	Concluido.
CP-038	Verificar que el botón de volver regresa a la página de selección de equipos.	El usuario hace clic sobre el botón volver.	Se muestra la página de equipos.	Se muestra la página de equipos.	Concluido.

Tabla 18. Pruebas CU-020 Visualizar plantillas liga.

Con estas pruebas quedan verificados todos los requisitos funcionales que se obtuvieron en la fase de análisis.

Los requisitos no funcionales se han comprobado uno por uno para verificar que hayan quedado plasmados en el sistema. Los requisitos de sistema deben cumplirse para empezar a desarrollar ya que definen las herramientas, tecnologías y versiones que se utilizarán para ello.

Los requisitos de interfaz se verifican durante la fase de desarrollo ya que sirven como guía para crear las diferentes maquetaciones de los portlets.

En cuanto a los requisitos de usabilidad se ha verificado que se hayan aplicado las medidas planteadas obtenidas del libro de patrones [40], aunque no se puede verificar que sean efectivas ya que no se han realizado pruebas con grupos de usuarios que aporten feedback para poder asegurarlo.

A parte de esto, al finalizar la fase de desarrollo se han vuelto a revisar todos los requisitos de la fase de análisis uno por uno para asegurar que no haya habido algún olvido de última hora.

En un principio los casos de uso y requisitos de software relacionados con la gestión de usuarios (registro, login, etc.) y la gestión de páginas se añadieron como trabajo a realizar, pero finalmente ni se diseñaron ni se codificaron, y finalmente se han desechado, ya que es el propio Liferay el que se encarga de toda esta gestión, por lo tanto no se han realizado tampoco la pruebas correspondientes a dichos casos de uso.

6.4 Conclusión

Esta última fase de pruebas cierra el ciclo de vida del desarrollo de mi sistema. Únicamente faltarían la fase de puesta en marcha y explotación y la fase de mantenimiento, pero estas quedan fuera del contexto de este proyecto. En el siguiente capítulo se realizará el presupuesto, exponiendo el tiempo que ha llevado cada fase y el coste de este.

Capítulo 7

Presupuesto

7.1 Introducción

Tras la finalización de todas las fases y una vez conocida la duración de todas ellas y el personal involucrado, se va a realizar el presupuesto del proyecto.

En primer lugar se realizará un resumen de la planificación del proyecto exponiendo su duración dividida por fases. A su vez se incluirá el diagrama de Gantt que define gráficamente como ha sido el desarrollo del proyecto dividiendo cada una de las fases en actividades más específicas. A continuación, se explica el coste supuesto de la realización de todo el proyecto desglosado en los diferentes tipos de costes que conforman el coste total. Finalmente se muestra una plantilla que resume todos los costes comentados.

Para realizar este presupuesto me he basado en la plantilla que ofrece la universidad como guía.

7.2 Calendario laboral

Se ha utilizado un calendario laboral estándar. Los días laborables son de lunes a viernes y se ha considerado un horario de 4 horas de trabajo al día. Como un mes tiene de media 22 días laborales se obtiene que un mes tiene 88 horas laborales.

7.3 Planificación del proyecto

El proyecto tiene una duración total de 7 meses y medio, o lo que es lo mismo 163 días laborales. Debido a que cada día laboral consta de 4 horas de trabajo se obtiene una cantidad de 752 horas de trabajo incluyendo el tiempo dedicado a la documentación.

Este tiempo se ha dividido en diferentes fases. Para la realización del proyecto se ha seguido el ciclo de vida típico de un proyecto informático con sus fases características. Estas fases son las siguientes:

Fase de reconocimiento del problema y estudio.

Esta fase es la más larga de todo el proyecto por la dificultad de tener que aprender gran cantidad de nuevos conceptos. En ella se incluye el estudio del problema a resolver, así como la búsqueda de posibles soluciones similares para hacerse una idea general. Una vez se tiene una visión clara, se estudian las tecnologías, herramientas y librerías que se usan para el desarrollo. También se incluye dentro de esta fase un periodo de aprendizaje del lenguaje Java y la instalación del entrono de desarrollo.

Análisis

En la fase de análisis se recogen las características y restricciones que la nueva aplicación debe tener. De esta fase se obtienen los documentos de casos de uso y de requisitos de software.

Diseño

En la fase de diseño se define como organizar los datos obtenidos en la fase de análisis de forma adecuada para su posterior ejecución. Se incluye la arquitectura del sistema, el diseño estático y dinámico de los diferentes componentes y el diseño de la base de datos que se utiliza.

Desarrollo

Es la fase de programación, lo obtenido en la fase de diseño se lleva a código. En ella se desarrollan los diferentes componentes que forman el portal.

Pruebas

En la fase de pruebas se realizan las pruebas del plan de pruebas para confirmar, detectar y corregir posibles errores en la fase de desarrollo.

Documentación

La fase de documentación incluye la realización de la memoria y toda la documentación que se ha obtenido durante las otras fases del proyecto. La realización de la memoria consiste en reunir toda esa información y organizarla para poder verla de forma clara.

En la siguiente tabla se expone el tiempo que ha llevado cada fase.

Fases	Días	Horas/Día	Horas totales
FASE DE ESTUDIO	65	4	260
FASE DE ANÁLISIS	12	4	48
FASE DE DISEÑO	18	4	72
FASE DE DESARROLLO	35	4	140
FASE DE PRUEBAS	10	4	40
FASE DE DOCUMENTACIÓN	163	1,18	192
		TOTAL	**752**

Tabla 19. Presupuesto - Horas dedicadas a cada fase.

A continuación se muestra el diagrama de Gantt que muestra de forma gráfica como ha sido el desarrollo de todo el proyecto dividiendo cada fase en tareas más específicas.

7.4 Diagrama de Gantt

	Nombre de tarea	Duración	Comienzo	Fin
1	**PROYECTO**	**163 días?**	**lun 13/10/14**	**mié 27/05/15**
2	**Estudio**	**65 días**	**lun 13/10/14**	**vie 09/01/15**
3	Estudio del problema a resolver	4 días	lun 13/10/14	jue 16/10/14
4	Java	10 días	vie 17/10/14	jue 30/10/14
5	Portal Liferay	6 días	vie 31/10/14	vie 07/11/14
6	Potlets	6 días	lun 10/11/14	lun 17/11/14
7	RSS	3 días	mar 18/11/14	jue 20/11/14
8	WS SOAP	8 días	vie 21/11/14	mar 02/12/14
9	WS REST	8 días	mié 03/12/14	vie 12/12/14
10	Twiter4Java	2 días	lun 15/12/14	mar 16/12/14
11	Google feed RSS	2 días	mié 17/12/14	jue 18/12/14
12	EJB	6 días	vie 19/12/14	vie 26/12/14
13	AJAX	2 días	lun 29/12/14	mar 30/12/14
14	Instalación de entorno de desarrollo	8 días	mié 31/12/14	vie 09/01/15
15	**Análisis**	**12 días**	**lun 12/01/15**	**mar 27/01/15**
16	Requisitos	6 días	lun 12/01/15	lun 19/01/15
17	Casos de uso	6 días	mar 20/01/15	mar 27/01/15
18	**Diseño**	**18 días**	**mié 28/01/15**	**vie 20/02/15**
19	Arquitectura del sistema	4 días	mié 28/01/15	lun 02/02/15
20	Diseño estático	6 días	mar 03/02/15	mar 10/02/15
21	Diseñoo dinámico	6 días	mié 11/02/15	mié 18/02/15
22	Diseño de Base de datos	2 días	jue 19/02/15	vie 20/02/15
23	**DESARROLLO**	**35 días**	**lun 23/02/15**	**vie 10/04/15**
24	EJB Cliente Mundial 2014	2 días	lun 23/02/15	mar 24/02/15
25	Portlets Mundial 2014	8 días	mié 25/02/15	vie 06/03/15
26	Hook(Listener)	4 días	lun 09/03/15	jue 12/03/15
27	Portlet Lector Twitter	5 días	vie 13/03/15	jue 19/03/15
28	Portlet lector RSS	4 días	vie 20/03/15	mié 25/03/15
29	Servicio Web LFP	3 días	jue 26/03/15	lun 30/03/15
30	EJB Cliente Liga LFP	2 días	mar 31/03/15	mié 01/04/15
31	Portlets Liga LFP	7 días	jue 02/04/15	vie 10/04/15
32	**PRUEBAS**	**10 días**	**lun 13/04/15**	**vie 24/04/15**
33	Diseño de pruebas	4 días	lun 13/04/15	jue 16/04/15
34	Relaización pruebas	6 días	vie 17/04/15	vie 24/04/15
35	**DOCUMENTACIÓN**	**163 días?**	**lun 13/10/14**	**mié 27/05/15**
36	Documentación	140 días?	lun 13/10/14	vie 24/04/15
37	Memoria	23 días	lun 27/04/15	mié 27/05/15

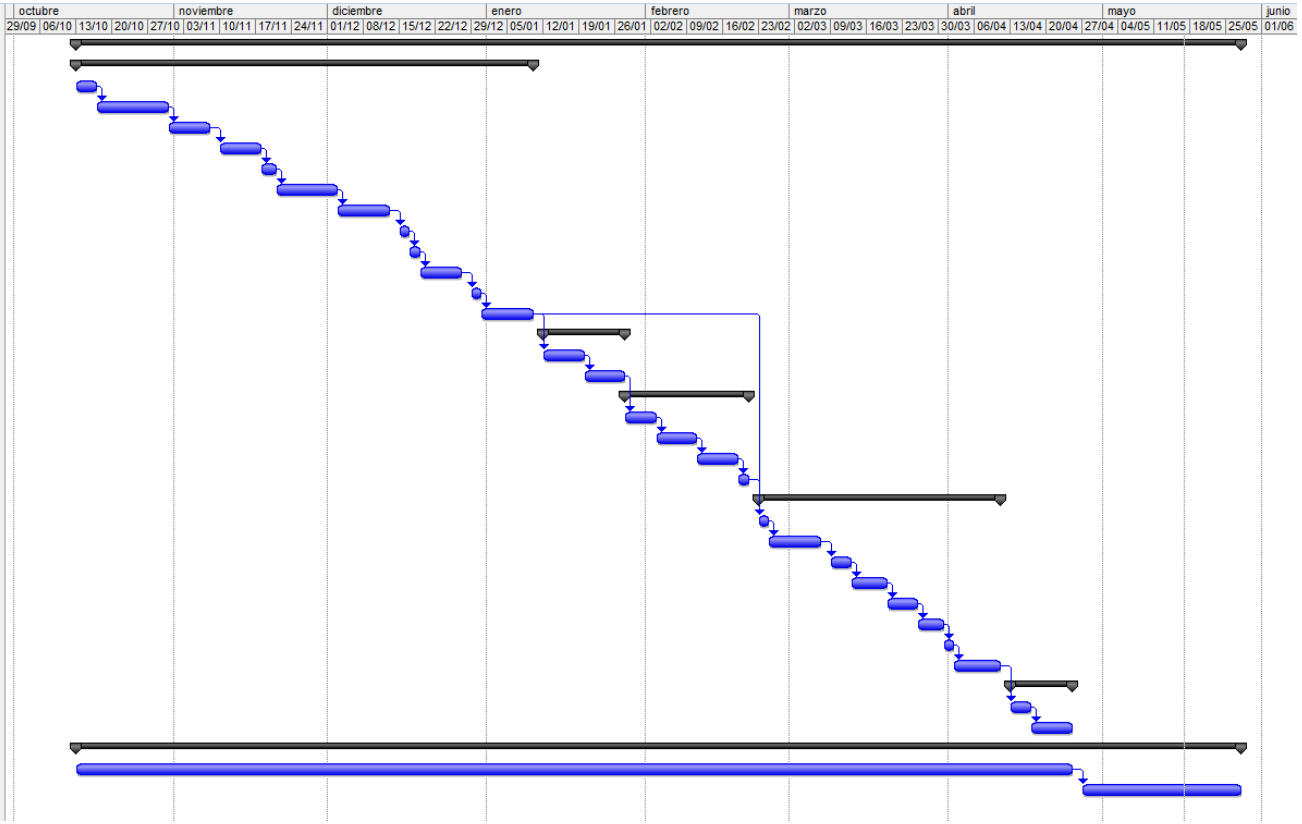

Ilustración 63. Diagrama de Gantt.

7.5 Costes

El precio total del proyecto se calcula obteniendo previamente los diferentes costes relacionados con este, coste de personal, de hardware, software y material fungible y añadiéndole un 20% de costes indirectos.

7.5.1 Coste de personal

Para la realización de las diferentes fases del proyecto se ha decidido incluir tres roles diferentes. El analista se encargará de la fase de análisis, es decir la obtención de los casos de uso y los requisitos, así como de la fase de diseño. El programador se encargará de la fase de estudio, de la de desarrollo y la documentación y el tester será el encargado de diseñar el plan de pruebas y realizarlas. Se han considerado los tres roles como junior al tener menos de 2 años de experiencia en este tipo de proyectos.

En la siguiente tabla se puede observar la carga de trabajo y el coste de cada recurso de personal por hora, obteniendo finalmente la suma total. Se ha considerado que cada mes tiene 88 horas laborables.

Personal	Horas	Precio/Hora	Dedicación(Hombre/mes)	Coste Total €
Analista Junior	120	17,94	1,37	2152,8
Programador Junior	592	13,46	6,72	7968,32
Tester Junior	40	9,69	0,46	387,6
				10508,72€

Tabla 20. Presupuesto - Coste de personal.

Los precios hora de cada rol los he obtenido informándome en la red sobre diferentes precios en diversas comunidades autónomas españolas y haciendo una estimación con el objetivo de obtener el precio para trabajadores junior. El coste de personal asciende a 10508.72€. Precio sin IVA.

7.5.2 Coste de Hardware.

En el coste de hardware se incluye todo el material que se ha utilizado para el desarrollo del proyecto calculando su amortización mediante la fórmula dada en la plantilla que ofrece la UC3M como guía.

$$\frac{A}{B} xCxD$$

A: Número de meses desde la fecha de facturación en que el equipo es utilizado.
B: Periodo de depreciación = 60 meses.
C: Coste del equipo.
D: Porcentaje del uso que se le dedica al proyecto.

En esta tabla se recogen los datos necesarios para aplicar la formula. En la última columna se recoge el coste que se ha obtenido de cada elemento de hardware, así como el coste total.

Hardware	Uds.	Coste (sin IVA)	Dedicación	Coste imputable
Ordenador Mountain GTM17 Ivy	1	1343	2,5 meses	55,95833333
Ordenador Toshiba Satellite A300	1	632	5 meses	52,66666667
Ratón óptico Logitech	1	19,75	7,5 meses	2,46875
Pendrive Lexar 16Gb	1	4,74	7,5 meses	0,5925
Disco Duro WD 500 GB	1	35,55	7,5 meses	4,44375
Impresora HP Laser	1	197,5	1 mes	3,291666667
				119,4216667€

Tabla 21. Presupuesto - Coste de hardware.

Tras realizar todos los cálculos se obtiene un coste de hardware que asciende a 119.42€. Precio sin IVA.

7.5.3 Coste de Software.

En este apartado se incluye el coste de los elementos de software utilizados en el proyecto, herramientas, librerías y editores de texto. Debido a que se ha procurado que todo el software fuera de código libre la mayoría de elementos tienen coste cero.

Software	Unid	Coste unidad (Sin IVA)	Coste Total
Netbeans	1	0	0
GlassFish Server	1	0	0
Liferay Portal	1	0	0
MySQL Server	1	0	0
Twitter4J	1	0	0
Google Feed API	1	0	0
JQuery	1	0	0
JQuery UI	1	0	0
Dia	1	0	0
Star UML	1	0	0
Microsoft Office 2007	1	177,75	177,75
Microsoft Project 2007	1	118,5	118,5
			296,25€

Tabla 22. Presupuesto - Coste de software.

La suma total asciende a 296.25€. Precio sin IVA.

7.5.4 Coste de material fungible.

Para terminar se incluye el coste de los materiales fungibles, es decir aquellos que se desgastan al usarlos.

Material fungible	Uds.	Coste unidad(Sin IVA)	Coste Total
Libreta	1	4,1	4,1
Tóner impresora (recarga)	2	39,5	79
Lápiz	1	0,2	0,2
Bolígrafo	1	0,2	0,2
Goma	1	0,15	0,15
Archivador	1	2,3	2,3
			85,95€

Tabla 23. Presupuesto - Coste de material fungible.

Tras hacer los cálculos se obtiene un coste de 85.95€. Precio sin IVA.
Una vez obtenidos todos los costes parciales se obtiene el coste total del proyecto.

7.5.5 Coste total.

La siguiente tabla muestra los costes obtenidos hasta ahora y le añade un 20% de costes indirectos.

Costes	Euros
Coste de personal	10508,72
Coste de material	119,42
Coste de Software	296,25
Coste de material fungible	85,95
Costes indirectos (20%)	2202,06
	13212,4€

Tabla 24. Costes totales.

Todos los costes se han calculado sin IVA por lo que el precio total del proyecto es de 13212.5€ sin IVA. A continuación se muestra la plantilla resumen con todos los costes calculados.

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:
Miguel Vidal Hernando
2.- Departamento:
Informática
3.- Descripción del Proyecto:
- Titulo: **Creación de un portal de sindicación de noticias.**
- Duración (meses): **7,5**
Tasa de costes Indirectos: 20%

4.- Presupuesto total del Proyecto (valores en Euros):
Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
Miguel Vidal Hernando		Analista Junior	1,37	1.578,72	2.152,80	
Miguel Vidal Hernando		Programador Junior	6,72	1.184,48	7.968,32	
Miguel Vidal Hernando		Tester Junior	0,46	852,72	387,60	
					0,00	
					0,00	
		Hombres mes	**8,55**	**Total**	10.508,72	

[a] 1 Hombre mes = 88 horas.

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Ordenador Mountain GTM17 Ivy	1343	100	2,5	60	55,96
Ordenador Toshiba Satellite A300	632	100	5	60	52,67
Raton óptico logitech	19,75	100	7,5	60	2,47
Pendrive Lexar 16Gb	4,74	100	7,5	60	0,59
Disco Duro WD 500 GB	35,55	100	7,5	60	4,44
Impresora HP Laser	197,5	100	1	60	3,29
				Total	119,42

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
Software		296,25
Material fungible		85,95
	Total	382,20

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,...

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	10.509
Amortización	119
Subcontratación de tareas	0
Costes de funcionamiento	382
Costes Indirectos	2.202
Total	13.212

El presupuesto total de este proyecto asciende a la cantidad de TRECEMIL DOSCIENTOS DOCE EUROS.

Colmenarejo a 27 Mayo de 2015
El ingeniero proyectista.

Fdo. Miguel Vidal Hernando.

Capítulo 8

Conclusiones y trabajo futuro

8.1 Introducción

Una vez finalizado el desarrollo del proyecto y terminada la memoria, en este capítulo se van a exponer las conclusiones obtenidas tras su realización. Además se van a describir las principales dificultades surgidas durante el desarrollo de cualquiera de las fases que han compuesto el proyecto y finalmente se comentarán algunas ideas sobre trabajos futuros que podrían llevarse a cabo.

8.2 Conclusiones

Como se comentó en el capítulo 1 de esta memoria, el propósito de este proyecto consistía en la creación de un portal de sindicación de información deportiva que obtuviera información de varias fuentes web y la reuniera para ofrecerla de forma centralizada a los usuarios.

Uno de los objetivos principales que se propusieron al comienzo del proyecto era desarrollar el contenido del portal como componentes modulares. Con el uso de las tecnologías definidas como objetivo (servicios web SOAP, servicios web REST y RSS) se desarrollaron una serie de portlets que conjuntamente conforman el contenido completo del portal.

Un portal como este aporta una forma diferente de informarse en Internet, mucho más personalizada y centralizada, ya que se puede obtener información de diferentes fuentes, diarios online, redes sociales, etc. sin tener que estar navegando por todos ellos, por lo que se gana mucho en rapidez y comodidad.

Además, al desarrollar el contenido con portlets, se le proporcionan al usuario una serie opciones de personalización tal como añadir y eliminar la información que desee o como la posibilidad de cambiar de lugar los portlets que tiene desplegados.

Debido a que es un portal de información, las funcionalidades que se le ofrecen al usuario no son muchas ya que no es el objetivo de este tipo de páginas. La verdadera importancia esta en ofrecer información actualizada. Con el uso de las tecnologías comentadas ya se cumple este objetivo, el portal obtiene la última información publicada y la muestra en la interfaz. En la mayoría de páginas que ofrecen información, como los diarios online, es necesario actualizar la página para poder ver nuevas noticias, por otro lado, estas no aparecen tan periódicamente como para que esto tenga gran importancia. Sin embargo en el portal desarrollado, se ofrece información proveniente de Twitter, que es una información que se actualiza mucho más a menudo y no tendría sentido tener que actualizar constantemente la página para poder leer nuevos tweets. Por este motivo se han implementado medidas para que el contenido se actualice sin tener que recargar la página. Para ello se han utilizado tecnologías como JavaScript y librerías como JQuery que permiten modificar elementos de la maquetación en tiempo real.

El último de los objetivos que se plantearon surgió de un posible problema que se me ocurrió antes de empezar a desarrollar los portlets, y era el tamaño variable de estos, ya que podría ocurrir que la maquetación creada se optimizara a un tamaño y no a otro dependiendo de donde estuviera localizado cada uno. Este problema se hizo realidad cuando terminé el primer portlet, por este motivo se desarrollaron algunos métodos para que los portlets con mayores problemas de maquetación se ajustaran y mostraran la información optimizada al tamaño del portlet en cada momento. Este cambio al igual que la actualización de la información se realiza en tiempo real debido a que los usuarios pueden cambiar los portlets de lugar en el momento que ellos deseen.

Tras haber repasado los objetivos generarles que se plantearon al inicio del proyecto se puede concluir que todos ellos se han cumplido.

También quiero comentar brevemente algunos aspectos sobre los conocimientos que he adquirido durante la realización de este proyecto.

En primer lugar quiero hacer referencia a que realizar un proyecto como este me ha ofrecido una visión mucho más cercana a lo que es desarrollar un proyecto en el ámbito profesional. Durante la carrera he realizado gran cantidad de prácticas muy interesantes y educativas, pero tras la realización del proyecto he sido realmente consciente de lo que es desarrollar un proyecto completo realizando la mayoría de las fases de desarrollo desde el inicio hasta el final. Ha resultado gratificante poder aplicar lo aprendido durante toda la carrera.

En cuanto a aspectos técnicos, he aprendido muchos nuevos conceptos relacionados con herramientas, tecnologías, programación etc. que expongo a continuación:

- Java: La realización del proyecto me ha servido para aprender el leguaje de programación Java que apenas conocía. Antes de comenzar con el proyecto estudié este lenguaje durante varios días y tras estos he ido aprendiendo nuevos conceptos a la vez que iba desarrollándolo. Una vez terminado considero que me he familiarizado con el uso de Java y la programación orientada a objetos y tengo una buena base para continuar aprendiendo nuevos aspectos sobre él.
- Liferay: He aprendido nuevos conceptos sobre Liferay Portal. Aprendí el concepto de portal durante la fase de estudio y posteriormente, debido a que no es una tarea sencilla instalarlo y configurarlo he adquirido conocimientos sólidos en este aspecto. También he aprendido que existe una amplia comunidad de desarrolladores muy activa, por lo que es una buena oportunidad de seguir aprendiendo sobre él con vistas a una posible salida profesional.
- Portlets: El aprendizaje de Liferay Portal conlleva el estudio y aprendizaje sobre portlets. He adquirido conocimientos sobre su ciclo de vida, y conceptos sobre su configuración y desarrollo.
- Servicios web Basados en SOAP: Sobre servicios web no tenía ningún conocimiento previo. En el mismo libro con el que aprendí Java, uno de los capítulos era sobre servicios web. En este libro aprendí sobre dos aspectos, los proveedores de servicio y los clientes que los consumen. Durante el desarrollo del proyecto he creado un servicio web basado en SOAP que ofrece información sobre la liga de fútbol profesional y varios portlets que lo consumen por lo que tengo una visión completa sobre ambos aspectos.
- Servicios web basados en REST: Tampoco tenía conocimientos sobre este tipo de servicio web y tras terminar el proyecto considero que he adquirido conocimientos necesarios para utilizarlos y a su vez la diferencia, ventajas y desventajas que ofrecen respecto a los basados en SOAP.
- RSS: Antes de comenzar el proyecto tenía algún conocimiento sobre el funcionamiento de RSS en general, en el rol de usuario ya que había utilizado

algún lector RSS previamente. Lo que no conocía era la estructura del formato en particular y debo comentar que una vez terminado el proyecto he aprendido lo necesario sobre éste como para poder manejar la información, seleccionando los elementos de cada ítem que me interesaban para crear mi propio lector RSS.

- EJB: Aunque no era una de las tecnologías planteadas como objetivo, he estudiado EJB como solución para reutilizar código a la hora de crear los portlets que consumían un servicio web basado en SOAP.
- AJAX: Al igual que EJB no era una de las tecnologías marcadas como objetivo del proyecto, pero actualmente hay pocas aplicaciones web que no la utilicen. Por este motivo decidí estudiar los conceptos básicos sobre ella con el objetivo de utilizarlos en algunos de los portlets desarrollados. Considero que he adquirido unos conocimientos básicos para entender cómo funciona y que me permiten utilizar algunas de las librerías que ofrecen funciones AJAX.

8.3 Dificultades encontradas.

Durante la realización del proyecto me he encontrado con numerosas dificultades y problemas que no quiero dejar de comentar brevemente para poder hacerse una idea del esfuerzo personal que me ha supuesto su resolución.

Una de las principales dificultades con las que me encontré durante la realización del proyecto fue con respecto a la instalación del entorno de desarrollo. En un principio obtuve las versiones estables más modernas de los elementos software, pensando que de esta forma tendría menos problemas, pero me encontré con todo lo contrario. A la hora de instalar lo elementos y ponerlos en funcionamiento no funcionaban correctamente. En ocasiones la versión del servidor de aplicaciones GlassFish no era compatible con la de Liferay Portal por lo que no arrancaban los servidores. En otra ocasión La versión del IDE no era compatible con la versión del plugin para crear y desplegar portlets. Los problemas de compatibilidad entre los distintos elementos me dificultaron en gran medida la instalación del entrono de desarrollo completo y fue una de las tareas que más tiempo me llevo hasta encontrar las versiones exactas compatibles entre sí.

Otro de los problemas más importantes fue el uso y configuración de los EJB utilizados. Durante la fase de estudio empecé a hacer mis pruebas con el servidor Jboss Portal en vez de GlassFish que al final se utilizó. El cambio de servidor de aplicaciones fue debido precisamente a problemas de funcionamiento con los EJB. No fui capaz de hacer la integración de las interfaces del EJB en el cliente ni por inyección de dependencia mi usando JNDI por lo que decidí cambiar de servidor por si ese era el problema. Al cambiar de servidor pude resolver este error, no sin antes tener que resolver otro problema con la manera de acceder a los EJB, mediante su interfaz local o a través de la remota. En un principio pensaba que la interfaz local era para acceder desde el mismo equipo y la remota desde otro equipo diferente. Esto no es así. La interfaz local es para acceder desde la misma máquina virtual de Java, mientras que la remota es para acceder desde fuera de esta. Como el EJB estaba dentro de un EAR y los portlets que

accedían estaban fuera, el acceso a través de la interfaz local no funcionaba. El problema lo solucioné cuando traté de acceder a través de la interfaz remota después de bastante tiempo pensando que el problema estaría en la inyección de dependencias o en el JNDI .

Tuve un problema a la hora de desarrollar el portlet del lector RSS. El lector que he creado ofrece diferentes fuentes de información para que el usuario pueda seleccionar la que el desee. El problema surgía debido a que algunos de los formatos RSS de los diferentes medios eran ligeramente diferentes unos de otros, por lo que tuve que tener estas pequeñas diferencias en cuenta a la hora de obtener los datos que mostraría en la interfaz.

Con el portlet lector de Twitter también tuve algún problema. En un principio me funcionaba correctamente, pero de un día para otro me apareció un mensaje de error en el log del servidor y me dejo de funcionar. Tarde varios días en darme cuenta del problema ya que no había modificado nada. Buscando información en foros acerca de cuál podía ser el problema, encontré que Twitter había cambiado los certificados de seguridad que usan los desarrolladores a la hora de conectarse a través de los servicios web REST que ofrece, por lo que tuve que actualizar los certificados de Liferay y realizar la conexión añadiendo la opción de acceder a través de SSL. Tras cambiar estos aspectos el portlet volvió a funcionar correctamente.

El desconocimiento inicial al desarrollar los portlets me llevó a cometer algunos errores que tuve que corregir una vez a avanzado el proyecto. Yo estaba acostumbrado a crear librerías JavaScript y archivos CSS como se realiza normalmente, una vez creados se incluye el script a utilizar y de ese modo se pueden utilizar sin problemas. Esto lo hice para cada portlet sin tener en cuenta que, si se hace de este modo, Liferay no tiene en cuenta que archivo pertenece a cada portlet, por lo que si añaden nombres iguales a funciones, variables o estilos de CSS, se producen errores al no saber que funciones y que estilos utilizar en cada portlet. Cuando empecé a obtener errores que no comprendía busque en foros cuál podía ser el problema y aprendí que para incluir este tipo de archivos en Liferay hay que hacerlo en un archivo de configuración general, añadiendo un contenedor a cada archivo con un nombre especifico, de esta manera se evitan problemas con nombres repetidos.

Por último también quiero comentar que al no conocer bien el leguaje Java al principio me resulto complicado el comenzar a programar, surgiendo los típicos errores obtenidos al comenzar con un lenguaje nuevo y más aún teniendo en cuenta que nunca había programado antes con un lenguaje orientado a objetos.

8.4 Trabajo futuro.

En este apartado expondré alguna de las ideas que me han surgido en cuanto a añadidos o posibles trabajos que podrían realizarse sobre el trabajo ya realizado en este proyecto.

- Un portal sindicador recoge información de distintas fuentes web para ofrecerlas de forma centralizada. En este proyecto se ha obtenido información utilizando

diversas tecnologías, pero se podían buscar y estudiar más para obtener información de otras fuentes, por ejemplo otras redes sociales como Facebook, Instagram, etc.

- La página resulta entretenida y útil para informarse de una forma centralizada y rápida sobre las últimas noticias deportivas del momento, y podría tener un gran número de visitas si se pusiera en producción. En ese caso habría que estudiar en profundidad el tema de los derechos legales, permisos o licencias, debido a que se está obteniendo información de otros medios y podría haber problemas legales en el caso de simplemente subirlo sin más.

- En servicio web sobre la liga de fútbol de primera división que he creado está localizado en un entrono local. Buscando por internet no he encontrado ninguna fuente que me ofreciera información sobre la liga por lo que podría ser una buena idea ofrecerlo en la red para que otras personas pudieran hacer uso de él, consumiendo los datos que se ofrecen.

- Optimizar la aplicación para poder usarse en dispositivos móviles utilizando algunas de las herramientas que ofrece Liferay, como por ejemplo la aplicación Device Recognition Provider [43], la cual permite a Liferay detectar qué dispositivo móvil o sistema operativo se utiliza para cualquier solicitud dada y altera el diseño de páginas basado en el dispositivo detectado.

Glosario

AJAX	*Asynchronous JavaScript And XML*
API	*Application Programming Interface*
CDDL	*Common Development and Distribution License*
CORBA	*Common Object Request Broker Architecture*
CSS	*Cascading Style Sheets*
DAO	*Data Access Object*
DCOM	*Distributed Component Object Model*
DOM	*Document Object Model*
EAR	*Enterprise Archive*
EJB	*Enterprise Java Bean*
FTP	File Transfer Protocol
GIF	*Graphics Interchange Format*
GPL2	*General Public License*
HTML	*HyperText Markup Language*
HTTP	*Hiper Text Transport Protocol*
IDE	Integrated development environment
JAX-WS	*Java API for XML Web Services*
JDBC	*Java DataBase Connectivity*
JDK	*Java Development Kit*
JNDI	*Java Naming and Directory Interface*
JPEG	*Joint Photographic Experts Group*
JRE	*Java Runtime Environment*
JSON	*JavaScript Object Notation*
JSP	*Java Server Pages*
JVM	*Java Virtual Machine*
LGPL	*Lesser General Public License*
MIME	*Multipurpose Internet Mail Extensions*
OAuth	*Open Authorization*
PNG	*Portable Network Graphics*
REST	*Representational state transfer*

RPC	*Remote Procedure Call*
RSS	Really Simple Syndication
SMTP	*Simple Mail Transfer Protocol*
SOA	*Service Oriented Architecture*
SOAP	*Simple Object Access Protocol*
SSL	*Secure Sockets Layer*
UDDI	*Universal Description, Discovery and Integration*
URI	*Uniform Resource Indentifier*
URL	*Uniform Resource Locator*
W3C	*World Wide Web Consortium*
WAR	*Web Application aRchive*
WSDL	*Web Services Description Language*
XML	*eXtensible Markup Language*

Referencias

1. Razón y palabra. [En línea] http://www.razonypalabra.org.mx/anteriores/n47/gomezpaniagua.html.

2. Marca.com. [En línea] http://www.marca.com/.

3. As.com. [En línea] http://as.com/.

4. Garmin Connect. [En línea] https://connect.garmin.com/es-ES/.

5. Acb.com. [En línea] http://www.acb.com/.

6. Netvibes. [En línea] http://www.netvibes.com/en.

7. **Perkins, Laura Lemay y Charles L.** *Aprendiendo Java 2 en 21 días.* s.l. : PHH, 1999. ISBN - 9789701702291. [Libro]

8. **Cadenhead, Rogers.** *Java 7.* s.l. : Anaya, 2012. ISBN - 978-84-415-3178-9. [Libro]

9. **Sarin, Ashish.** *Portlets in Action.* s.l. : Manning, 2012. ISBN - 9781935182542. [Libro]

10. Jboss portal. [En línea] http://jbossportal.jboss.org/.

11. Liferay. [En línea] https://www.liferay.com/.

12. **Hakim, Fady.** Portlet vs Widget. [En línea] https://www.liferay.com/es/web/fady.hakim/blog/-/blogs/portlet-vs-widget.

13. **w3schools.** RSS. [En línea] http://www.w3schools.com/webservices/rss_intro.asp.

14. Defensa Central. [En línea] http://www.defensacentral.com/.

15. **W3C.** Web Services Architecture. *Working Group Note 11 February 2004.* [En línea] 11 de Febrero de 2004. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/.

16. **Fielding, Roy Thomas.** Architectural Styles and the Design of Network-based Software Architectures. [En línea] 2000. https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf.

17. **Marset, Rafael Navarro.** REST vs Web Services. [En línea] 2006-2007. http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf.

18. **w3schools.** Web Services SOAP. [En línea] http://www.w3schools.com/webservices/default.asp.

19. **w3schools.** AJAX. [En línea] http://www.w3schools.com/ajax.

20. JSR 168: Portlet Specification. [En línea] https://www.jcp.org/en/jsr/detail?id=168.

21. JSR 286: Portlet Specification 2.0. [En línea] https://www.jcp.org/en/jsr/detail?id=286.

22. **Aransay, Alberto Los Santos.** Revisión de los servicios web SOAP/REST: Características y rendimiento. [En línea] Marzo de 2009. http://www.albertolsa.com/wp-content/uploads/2009/07/mdsw-revision-de-los-servicios-web-soap_rest-alberto-los-santos.pdf.

23. **Wladimir Rodríguez - Universidad de Los Andes - Mérida (Venezuela)-.** Arquitectura orientada a servicios. *Videos youtube (8 videos).* [En línea] 2013. https://www.youtube.com/watch?v=UJ3ZVoQitmw&index=1&list=PLcL8RDzOxvIrTVV-7G1TkNeyBc9oFmmhg.

24. **RSS Advisory Board.** RSS 2.0 Specification. [En línea] 30 de Marzo de 2009. http://www.rssboard.org/rss-specification.

25. Google Feed API. [En línea] https://developers.google.com/feed/?hl=es.

26. **Debu Panda, Reza Rahman y Derek Lane.** *EJB 3 in Action.* s.l. : Manning, 2007. ISBN 1-933988-34-7. [Libro]

27. **Sikora, Michael.** *EJB 3 Developer Guide.* s.l. : Packt Publishing Ltd., 2008. ISBN 978-1-847195-60-9. [Libro]

28. **w3schools.** JavaScript HTML DOM. [En línea] http://www.w3schools.com/js/js_htmldom.asp.

29. JQuery. [En línea] https://jquery.com/.

30. **Murphey, Rebecca.** Fundamentos de JQuery. [En línea][Libro] https://librosweb.es/libro/fundamentos_jquery/.

31. Netbeans 6.9.1 - Download. [En línea] https://netbeans.org/downloads/6.9.1/.

32. Netbeans Portalpack. [En línea] https://contrib.netbeans.org/portalpack.

33. JDK 6 - Dowload. [En línea] http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase6-419409.html.

34. GlassFish 3.0.1 - Download. [En línea] https://glassfish.java.net/es/downloads/3.0.1-final.html.

35. MySQL - Downloads. [En línea] http://www.mysql.com/downloads/.

36. Twitter4j. [En línea] http://twitter4j.org/en/index.html.

37. JQuery User Interface. [En línea] http://jqueryui.com/.

38. StarUML. [En línea] http://staruml.io/.

39. Instalación de Liferay en un servidor GlassFish v3 para producción. [En línea] http://www.apuntesdejava.com/2010/07/instalacion-de-liferay-en-un-servidor.html.

40. **Douglas K. van Duyne, James Landay y Jason Hong.** *The Design of Sites: Patterns for Creating Winning Websites.* s.l. : Prentice Hall, 2007. ISBN 978-0131345553. [Libro]

41. Comunio. [En línea] http://www.comunio.es/.

42. **Cadenhead, Rogers.** Crear servicios Web con JAX-WS. *Java 7.* s.l. : Anaya, 22, págs. 322 - 335. [Libro]

43. Device Recognition Provider EE. [En línea] http://www.liferay.com/es/marketplace/-/mp/application/35419014.

Apéndice A: Documento de casos de uso

Introducción

Propósito

El propósito del documento es recoger el conjunto de casos de uso para el diseño e implementación del portal automatizado de noticias deportivas y los actores que interaccionan con el sistema.

Alcance

Los casos de uso describen de forma detallada las funcionalidades que los usuarios pueden realizar en el sistema, teniendo en cuenta los pasos concretos necesarios para llevar a cabo la funcionalidad, tanto en los escenarios de éxito como en los posibles escenarios alternativos. Además se describirá las precondiciones y postcondiciones a tener en cuenta en cada funcionalidad concreta y qué tipo de usuario podrá llevarla a cabo.

Resumen

En la primera parte de este documento de identificarán y describirán los diferentes tipos de actores que formarán parte del sistema.

Una vez descritos los actores se podrá encontrar los diagramas de casos de uso en notación UML. Estos definen el comportamiento del sistema ante la interacción de los diferentes actores.

A continuación se realizará la descripción de forma extendida de cada caso de uso mediante tablas. En estas tablas se tendrán en cuenta los actores involucrados, precondiciones, postcondiciones y los escenarios de éxito y alternativos.

Por último se dispondrá la matriz de trazabilidad, que definirá la conexión existente entre los casos de uso y los requisitos de software.

Acrónimos

- **UML:** *Unified Modeling Language.*
- **RSS***: Really Simple Syndication.*

Definiciones

- **Tweet:** Cada uno de los mensajes de texto escritos por los usuarios de Twitter. Están compuestos por 140 caracteres como máximo.
- **Feed RSS:** Fichero de tipo XML generado por los sitios web para ofrecer una versión reducida de la información incluida en el sitio web. Este archivo está compuesto por ítems que constan de un titular, un resumen de la información, un enlace a la noticia completa, autor, fecha de publicación y en algunas ocasiones, una fotografía que ilustra la información. El fichero se reescribe cuando se actualiza la información en el sitio web. De esta forma es posible acceder a la información del sitio web sin necesidad de visitarlo.

Casos de uso

En los siguientes subapartados se van a describir los actores, los diagramas de casos de uso realizados en UML y los casos de uso de forma extendida.

Actores

Los actores no representan a usuarios concretos, sino a los diferentes roles mediante los cuales los usuarios van a poder interactuar con el sistema. Cada rol se compondrá de un conjunto limitado de acciones a realizar frente al sistema.

En la siguiente tabla se describen los diferentes roles del sistema:

Actor	Descripción
Visitante	Este actor representará a los usuarios que accedan al portal automático de noticias para informarse y que no tengan un usuario registrado en el portal. De esta manera todo visitante del portal podrá informarse de forma rápida y sencilla sin necesidad de tener un usuario registrado. Si lo desean podrán también crear un usuario en el sistema.
Usuario miembro	Este actor representará a los usuarios que posean un usuario

	registrado en el portal y que estén conectados. Estos usuarios podrán visualizar la información del portal además de gestionar su espacio privado en el portal.
Administrador	El usuario con este rol será el encargado de gestionar el portal, tanto la disposición de las columnas y los portlets como decidir que portlets aparecerán en la página principal. Además podrá visualizar la información del portal.

Tabla 25. Tabla de actores

El usuario administrador del portal funcionará como actor *Administrador* y como actor *Usuario miembro* al mismo tiempo, ya que además de gestionar el portal, también tendrá su espacio privado. El resto de usuarios registrados en el portal únicamente funcionará como *Usuario Miembro*.

Diagramas de casos de uso

A continuación se representará el diagrama de casos de uso de la aplicación. Este será descompuesto en subdiagramas más concretos para ofrecer un mayor detalle de las principales secciones.

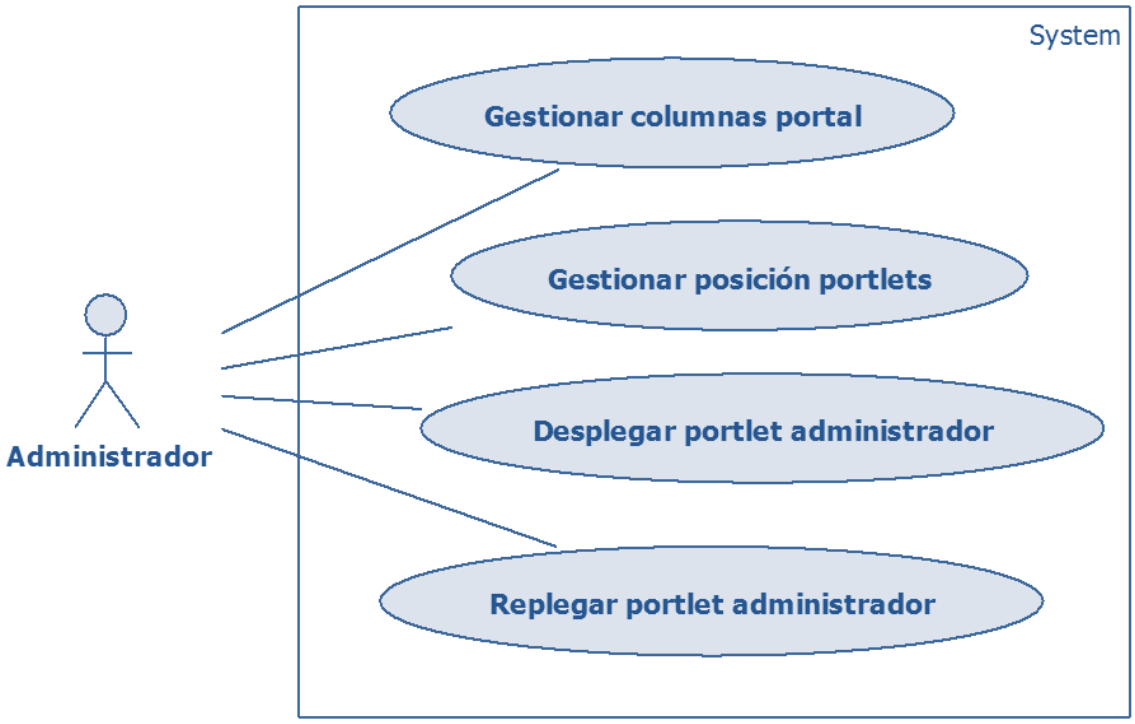

Ilustración 64. Diagrama de casos de uso general.

Para que tanto los usuarios miembros como el administrador puedan realizar sus correspondientes gestiones, ambos deben primero autenticarse en el portal con su usuario.

A continuación se muestran los casos de uso *visualizar información*, *gestionar espacio privado* y *gestionar portal* de forma más detallada.

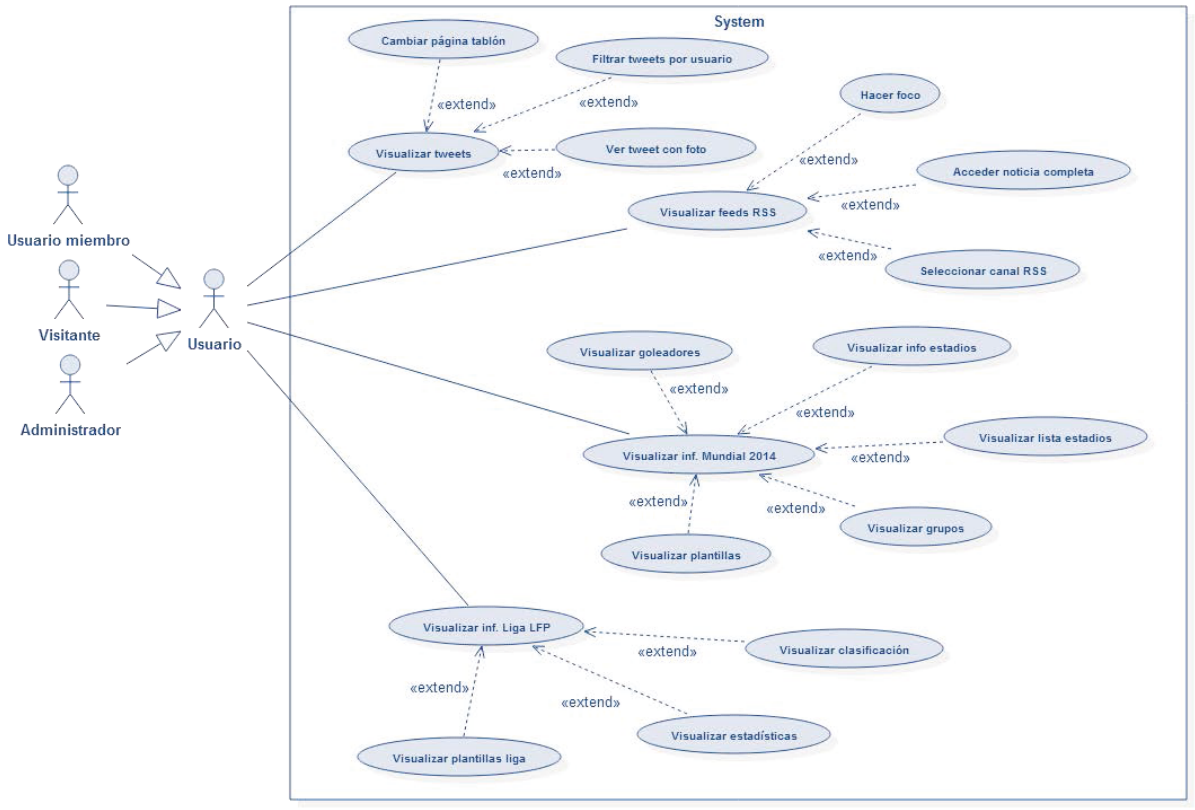

Ilustración 65. Diagrama de casos de uso: visualizar información.

Visualizar inf. Mundial 2014 siempre conllevará visualizar los 8 grupos del Mundial con sus correspondientes selecciones. Esto es así porque los grupos conforman la interfaz de entrada a la información del Mundial 2014..

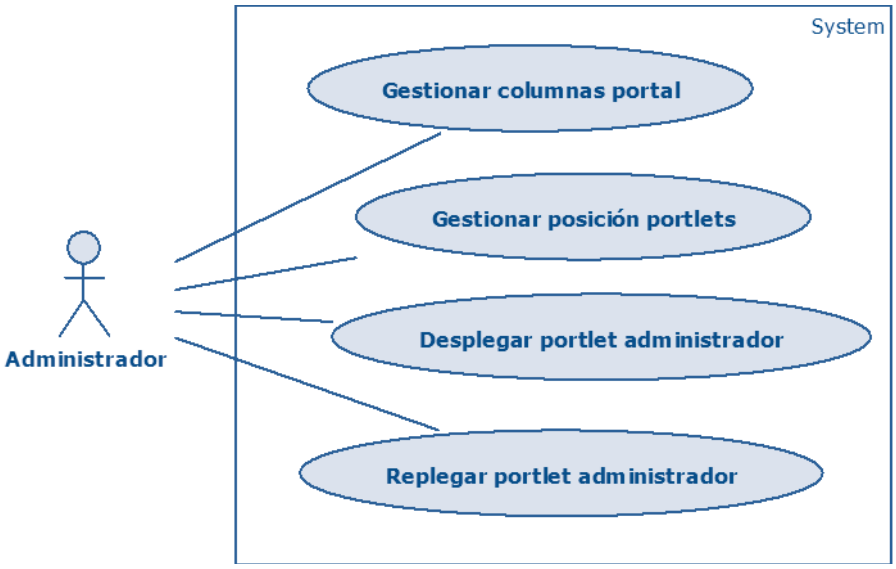

Ilustración 66. Diagrama de casos de uso: gestionar portal.

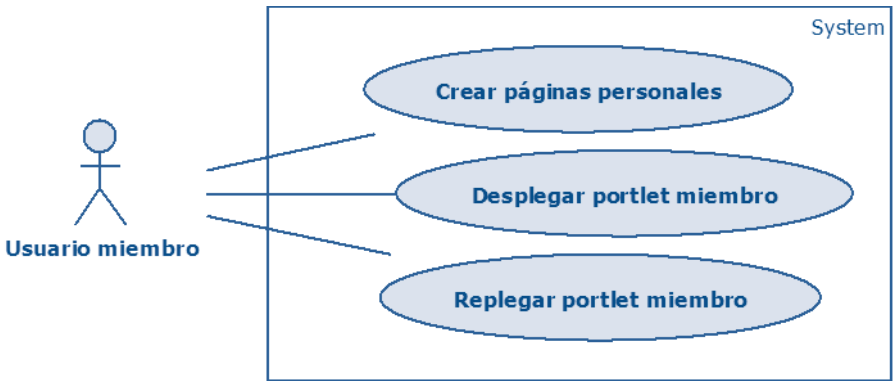

Ilustración 67. Casos de uso: gestionar espacio privado.

A continuación se mostrará la descripción extendida de cada caso de uso mediante tablas. Estas tablas incluyen en nombre del caso de uso, su identificador, los actores involucrados, su descripción, las precondiciones que deben darse, los escenarios de éxito y alternativos y la postconción o estado del sistema después de haberse producido el escenario indicado.

Casos de uso extendidos

A continuación se definen los casos de uso de forma extendida a partir de tablas.

CASO DE USO	Registrarse	
IDENTIFICADOR	CU-001	
ACTORES	Visitante.	
DESCRIPCIÓN	El visitante podrá crear una cuenta de usuario propia en el sistema.	
PRECONDICIÓN	-	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. El visitante accederá al formulario de registro. 2. Introducirá los datos en el formulario (nombre, segundo nombre, apellido, nombre de usuario, dirección de correo (ID), fecha de nacimiento, género y texto de verificación).	3. Comprobará que el texto de verificación es correcto y que la dirección de correo no esté repetida. 4. Creará el usuario en la base de datos.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
	1. El visitante accederá al formulario de registro. 2. Introducirá los datos en el formulario (nombre, segundo nombre, apellido, nombre de usuario, dirección de correo (ID), fecha de nacimiento, género y texto de verificación).	3. Comprueba el texto de verificación y no es correcto o la dirección de correo está repetida. 4. Mostrará el error correspondiente y no creará la entrada en la base de datos.
POSTCONDICIÓN	El visitante tendrá un usuario creado en el portal pero no estará autenticado en el sistema.	

Tabla 26. CU-001 Registrarse

CASO DE USO	Autenticarse	
IDENTIFICADOR	CU-002	
ACTORES	Usuario miembro, Administrador.	
DESCRIPCIÓN	Un visitante podrá autenticarse en el sistema.	
PRECONDICIÓN	Debe existir el correspondiente usuario registrado.	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. El visitante introducirá su identificador y contraseña en el formulario de autenticación y enviará los datos.	2. Comprobará que los datos introducidos son correctos. 3. Mostrará las opciones correspondientes del actor al que pertenece el usuario autenticado.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
	1. El visitante introducirá su identificador y contraseña en el formulario de autenticación y enviará los datos.	2. Comprobará los datos introducidos y estos son incorrectos. 3. Informará del error al usuario y no mostrará ninguna opción nueva.
POSTCONDICIÓN	El usuario quedará autenticado y tendrá las opciones del rol correspondiente.	

Tabla 27. CU-002 Autenticarse

CASO DE USO	Visualizar tweets	
IDENTIFICADOR	CU-003	
ACTORES	Visitante, Usuario miembro, Administrador.	
DESCRIPCIÓN	Un usuario podrá informarse a través de los tweets escritos por los usuarios seguidos por la cuenta de Twitter del portal.	
PRECONDICIÓN	-	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
		1. Obtendrá los diez tweets más modernos grabados en la base de datos, y los mostrará en el tablón. 2. Calculará el número de páginas necesarias para poder mostrar todos los tweets guardados en la base de datos y lo mostrará en un menú desplegable. 3. Mostrará un menú desplegable con todos los usuarios que están siendo seguidos por la cuenta de Twitter del portal.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
	1. El usuario Administrador en la página principal o un usuario miembro en su espacio privado seleccionarán el portlet de noticias vía RSS y lo cambiarán de posición a una columna de diferente anchura.	2. En la próxima actualización, recompondrá tanto la disposición de la noticia como en número de noticias a mostrar (10 noticias en las columnas más anchas y 5 en las más estrechas).
POSTCONDICIÓN	Se actualizará el tablón cada 15 segundos.	

Tabla 28. CU-003 Visualizar tweets

CASO DE USO	Cambiar página tablón	
IDENTIFICADOR	CU-004	
ACTORES	Visitante, Usuario miembro, Administrador	
DESCRIPCIÓN	Un usuario que estará observando la información mediante el portlet de Twitter, podrá informarse sobre los mensajes más antiguos.	
PRECONDICIÓN	-	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. Seleccionará en el menú desplegable de páginas el número de página que desee ver.	2. Calculará que mensajes deben mostrarse en la página seleccionada y los mostrará en el tablón.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
POSTCONDICIÓN	Continuará actualizándose cada 15 segundos.	

Tabla 29. CU-004 Cambiar página tablón

CASO DE USO	Filtrar tweets por usuario	
IDENTIFICADOR	CU-005	
ACTORES	Visitante, Usuario miembro, Administrador	
DESCRIPCIÓN	Un usuario que estará observando la información mediante el portlet de Twitter, podrá informarse sobre los mensajes de un usuario concreto.	
PRECONDICIÓN	-	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. Seleccionará en el menú desplegable de usuarios el nombre del usuario que le interese.	2. Obtendrá de la base de datos únicamente los mensajes del usuario seleccionado y los mostrará en el tablón. 3. Calculará el número de páginas necesarias para mostrar todos los mensajes del usuario seleccionado y actualizará el menú desplegable de páginas.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
POSTCONDICIÓN	Continuará actualizándose cada 15 segundos.	

Tabla 30. CU-005 Filtrar tweets por usuario

CASO DE USO	Ver tweet con foto	
IDENTIFICADOR	CU-006	
ACTORES	Visitante, Usuario miembro, Administrador	
DESCRIPCIÓN	Un usuario podrá visualizar la información de un tweet con foto en una ventana modal, para poder ver la foto a mayor tamaño.	
PRECONDICIÓN	Debe haber un tweet que contenga una foto.	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. Hará clic sobre la foto del tweet a agrandar.	2. Obtendrá la información del tweet y la mostrará en una ventana modal junto con la foto a mayor tamaño.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
	3. Se hará clic en el botón de cerrar	4. Cerrara la ventana modal.
POSTCONDICIÓN	-	

Tabla 31. CU-006 Ver tweet con foto

CASO DE USO	Visualizar feeds RSS	
IDENTIFICADOR	CU-007	
ACTORES	Visitante, Usuario miembro, Administrador	
DESCRIPCIÓN	Un usuario podrá informarse mediante la lectura de la información ofrecida por los feeds RSS de los diversos sitios de información deportiva.	
PRECONDICIÓN	-	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
		1. Obtendrá las últimas noticias del feed RSS del sitio web seleccionado, obteniendo la imagen de la noticia, el titular, el resumen y el enlace a la noticia completa. 2. Las mostrará mediante un slideshow en el portlet correspondiente.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
	1. El usuario Administrador en la página principal o un usuario miembro en su espacio privado seleccionarán el portlet lector de RSS y lo cambiarán de posición a una columna de diferente anchura.	2. En la próxima actualización, recompondrá la maquetación del lector de RSS.
POSTCONDICIÓN	El slideshow cambiará automáticamente el foco de una noticia a la siguiente en un bucle circular cada 7 segundos.	

Tabla 32. CU-007 Visualizar feeds RSS

CASO DE USO	Hacer foco	
IDENTIFICADOR	CU-008	
ACTORES	Visitante, Usuario miembro, Administrador	
DESCRIPCIÓN	Un usuario podrá parar el bucle automático del slideshow si desea ver una noticia en particular.	
PRECONDICIÓN	-	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. El usuario pasará el ratón por encima del titular de la noticia.	2. Detendrá el bucle del slideshow manteniendo la noticia seleccionada por el usuario.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
	1. El usuario quitará el ratón de encima del titular de la noticia.	2. Iniciará de nuevo el bucle del slideshow comenzando por la última noticia marcada por el usuario.
POSTCONDICIÓN	-	

Tabla 33. CU-008 Hacer foco

CASO DE USO	Acceder noticia completa	
IDENTIFICADOR	CU-009	
ACTORES	Visitante, Usuario miembro, Administrador	
DESCRIPCIÓN	Un usuario podrá leer la noticia completa si le interesa una vez leído el resumen.	
PRECONDICIÓN	-	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. El usuario pasará el ratón por encima del titular de la noticia. 3. El usuario pinchará sobre el titular que será un hipervínculo.	2. Detendrá el bucle del slideshow manteniendo la noticia seleccionada por el usuario. 4. Abrirá la noticia completa del sitio web de la fuente en una nueva pestaña.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
POSTCONDICIÓN	-	

Tabla 34. CU-009 Acceder noticia completa

CASO DE USO	Seleccionar canal RSS	
IDENTIFICADOR	CU-010	
ACTORES	Visitante, Usuario miembro, Administrador	
DESCRIPCIÓN	Un usuario podrá cambiar el canal RSS a visualizar por otro de los ofrecidos.	
PRECONDICIÓN	-	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. El usuario seleccionará del menú desplegable el nombre del medio de información que desee visualizar.	2. Detendrá todo el funcionamiento del portlet. 3. Cargará el nuevo canal RSS y pondrá el portlet en funcionamiento de nuevo.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
POSTCONDICIÓN	-	

Tabla 35. CU-010 Seleccionar canal RSS

CASO DE USO	Visualizar inf. Mundial 2014	
IDENTIFICADOR	CU-011	
ACTORES	Visitante, Usuario miembro, Administrador	
DESCRIPCIÓN	Un usuario desea informarse sobre el Mundial de Brasil 2014. Este podrá visualizar los grupos oficiales con sus correspondientes selecciones, las plantillas de las diferentes selecciones, Información sobre los estadios y los máximos goleadores. Este es un caso de uso abstracto ya que visualizar esta información siempre conllevará visualizar primero los grupos, debido a que esta será la interfaz de entrada a esta información. Por esta razón los escenarios de éxito y alternativo se describirán en los casos de uso CU-012, CU-013, CU-014, CU-015 y CU-016.	
PRECONDICIÓN	-	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
POSTCONDICIÓN	-	

Tabla 36. CU-011 Visualizar inf. Mundial 2014

CASO DE USO	Visualizar grupos	
IDENTIFICADOR	CU-012	
ACTORES	Visitante, Usuario miembro, Administrador	
DESCRIPCIÓN	Un usuario podrá informarse sobre el Mundial de Brasil 2014, viendo los grupos con sus correspondientes selecciones.	
PRECONDICIÓN	-	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
		1. Obtendrá los grupos del Mundial con sus correspondientes selecciones, mediante las clases java creadas a partir del fichero WSDL del servicio web. 2. Dispondrá la información en el portlet correspondiente.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
	1. El usuario habrá accedido a una de las selecciones y pinchará sobre el enlace de volver a los grupos.	2. El sistema volverá a cargar los grupos en el portlet.
POSTCONDICIÓN	-	

Tabla 37. CU-012 Visualizar grupos

CASO DE USO	Visualizar plantillas	
IDENTIFICADOR	CU-013	
ACTORES	Visitante, Usuario miembro, Administrador	
DESCRIPCIÓN	Un usuario podrá informarse sobre el Mundial de Brasil 2014. Desea analizar con que plantilla participaron las diferentes selecciones en el Mundial.	
PRECONDICIÓN	Deberá estar cargada la vista de grupos en el portlet.	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. El usuario pinchará sobre una de las selecciones.	2. Obtendrá los jugadores que conforman la selección elegida, mediante las clases java creadas a partir del fichero WSDL del servicio web. 3. Dispondrá la información en el portlet.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
POSTCONDICIÓN	-	

Tabla 38. CU-013 Visualizar plantillas

CASO DE USO	Visualizar lista estadios	
IDENTIFICADOR	CU-014	
ACTORES	Visitante, Usuario miembro, Administrador	
DESCRIPCIÓN	Un usuario que desee informarse sobre los estadios del Mundial de Brasil 2014, podrá acceder al listado de estadios.	
PRECONDICIÓN	-	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. El usuario, que se encuentra en una vista diferente a la del listado de estadios, seleccionará el enlace que le conducirá a este listado.	2. Cambiará la vista mostrada sobre el portlet por la del listado de estadios.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
	1. El usuario se encuentra en la vista del listado de estadios.	2. El sistema no ofrecerá el enlace al listado de estadios ya que esta vista ya está dispuesta en el portlet.
POSTCONDICIÓN	-	

Tabla 39. CU-014 Visualizar lista estadios

CASO DE USO	Visualizar info estadios	
IDENTIFICADOR	CU-015	
ACTORES	Visitante, Usuario miembro, Administrador	
DESCRIPCIÓN	Un usuario podrá acceder a la información de un estadio en concreto.	
PRECONDICIÓN	Deberá estar cargada la vista del listado de estadios en el portlet.	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. El usuario seleccionará uno de los estadios y clicará sobre su imagen.	2. Mostrará la información del estadio en el portlet. Esta información se obtendrá mediante las clases java creadas a partir del fichero WSDL del servicio web.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
	1. El usuario habrá accedido a uno de los estadios y clicará sobre el enlace de volver al listado de estadios.	2. El sistema volverá a cargar la vista del listado de estadios.
POSTCONDICIÓN	-	

Tabla 40. CU-015 Visualizar info estadios

CASO DE USO	Visualizar goleadores	
IDENTIFICADOR	CU-016	
ACTORES	Visitante, Usuario miembro, Administrador	
DESCRIPCIÓN	Un usuario podrá informarse sobre los máximos goleadores del Mundial de Brasil 2014.	
PRECONDICIÓN	-	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
		1. Obtendrá la información de los máximos goleadores del Mundial, mediante las clases java creadas a partir del fichero WSDL del servicio web. 2. Dispondrá la información en el portlet correspondiente.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
POSTCONDICIÓN	-	

Tabla 41. CU-016 Visualizar goleadores

CASO DE USO	Visualizar inf. Liga LFP	
IDENTIFICADOR	CU-017	
ACTORES	Visitante, Usuario miembro, Administrador	
DESCRIPCIÓN	Un usuario desea informarse sobre la liga de Futbol de primera división. Este podrá visualizar la clasificación de los equipos, las estadísticas en cuanto a goles, asistencias y amonestaciones de los jugadores y las plantillas de los diferentes equipos. Este es un caso de uso abstracto, los escenarios de éxito y alternativo se describirán en los casos de uso CU-018, CU-019 y CU-020.	
PRECONDICIÓN	-	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
POSTCONDICIÓN	-	

Tabla 42. CU-017 Visualizar inf. Liga LFP

CASO DE USO	Visualizar clasificación	
IDENTIFICADOR	CU-018	
ACTORES	Visitante, Usuario miembro, Administrador	
DESCRIPCIÓN	Un usuario podrá informarse sobre la clasificación de los equipos de primera división mediante una tabla ordenable con la información.	
PRECONDICIÓN	-	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
		1. Obtendrá la información correspondiente a la clasificación de los equipos mediante las clases Java creadas a partir del fichero WSDL del servicio web. 2. Dispondrá la información en el portlet correspondiente.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
	1. El usuario hará clic sobre una de las cabeceras de una columna para ver los datos ordenados.	2. El sistema mostrará los datos ordenados por dicha columna.
POSTCONDICIÓN	-	

Tabla 43. CU-018 Visualizar clasificación

CASO DE USO	Visualizar estadísticas	
IDENTIFICADOR	CU-019	
ACTORES	Visitante, Usuario miembro, Administrador	
DESCRIPCIÓN	Un usuario podrá informarse sobre las estadísticas de la liga LFP en cuanto a goles, asistencias y amonestaciones mediante una tabla ordenable. En principio se mostrará la tabla de goleadores, pudiendo el usuario cambiar de tabla si así lo desea.	
PRECONDICIÓN	-	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
		1. Obtendrá la información correspondiente a los goleadores mediante las clases Java creadas a partir del fichero WSDL del servicio web. 2. Dispondrá la información en el portlet correspondiente.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
	3. El usuario hará clic sobre cualquiera de los botones que le permitirán cambiar de tabla, ya sea asistencias o amonestaciones. 5. El usuario hará clic sobre una de las cabeceras de una columna para ver los datos ordenados.	4. El sistema mostrará la nueva tabla con los datos de asistencias o amonestaciones. 6. El sistema mostrará los datos ordenados por dicha columna.
POSTCONDICIÓN	-	

Tabla 44. CU-019 Visualizar estadísticas

CASO DE USO	Visualizar plantillas liga	
IDENTIFICADOR	CU-020	
ACTORES	Visitante, Usuario miembro, Administrador	
DESCRIPCIÓN	Un usuario podrá informarse sobre las plantillas de jugadores que componen cada equipo de primera división así como su información correspondiente a la competición.	
PRECONDICIÓN	Deberá estar cargada la vista de equipos en el portlet.	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. El usuario hará clic sobre una de los equipos.	2. Obtendrá los jugadores que conforman el equipo elegido, mediante las clases java creadas a partir del fichero WSDL del servicio web. 3. Dispondrá la información en el portlet.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
	1. El usuario hará clic sobre una de las cabeceras de una columna para ver los datos ordenados. 3. El usuario habrá accedido a una de las plantillas y clicará sobre el enlace de volver al listado de equipos.	2. El sistema mostrará los datos ordenados por dicha columna. 4. El sistema volverá a cargar la vista del listado de estadios.
POSTCONDICIÓN	-	

Tabla 45. CU-020 Visualizar plantillas liga

CASO DE USO	Gestionar columnas portal	
IDENTIFICADOR	CU-021	
ACTORES	Administrador	
DESCRIPCIÓN	El administrador podrá cambiar la disposición de las columnas del portal.	
PRECONDICIÓN	El usuario deberá estar autenticado como administrador.	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. El usuario administrador accederá en el menú superior al apartado de gestionar columnas. 3. El usuario seleccionará la disposición que desee y guardará los cambios.	2. El sistema mostrará las diferentes disposiciones disponibles. 4. El sistema modificará la disposición de columnas del portal.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
POSTCONDICIÓN	La nueva disposición se mantendrá mientras el usuario administrador no vuelva a cambiarla, aunque se cierre la sesión.	

Tabla 46. CU-021 Gestionar columnas portal

CASO DE USO	Gestionar posición portlets	
IDENTIFICADOR	CU-022	
ACTORES	Administrador	
DESCRIPCIÓN	El administrador podrá cambiar la posición de un portlet en el portal.	
PRECONDICIÓN	El portlet deberá estar desplegado en el portal y el usuario autenticado como administrador.	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. El administrador pinchará sobre el portlet y manteniéndolo pulsado lo arrastrará hasta la posición deseada. 3. El administrador soltará el portlet en la posición deseada.	2. El sistema mostrará una previsualización del portlet situado en la nueva posición. 4. El sistema modificará la posición final del portal.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
	1. El administrador pinchará sobre el portlet y manteniéndolo pulsado lo arrastrará hasta la posición deseada. 3. El administrador soltará el portlet en una posición no válida.	2. El sistema no mostrará la previsualización del portlet situado en la nueva posición debido a que la posición no es válida. 4. El sistema mantendrá el portlet en la posición inicial.
POSTCONDICIÓN	La nueva posición del portlet se mantendrá mientras el usuario administrador no vuelva a cambiarla, aunque se cierre la sesión.	

Tabla 47. CU-022 Gestionar posición portlets

CASO DE USO	Desplegar portlet administrador	
IDENTIFICADOR	CU-023	
ACTORES	Administrador	
DESCRIPCIÓN	El administrador podrá desplegar un nuevo portlet en el portal.	
PRECONDICIÓN	El usuario deberá estar autenticado como administrador.	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. El usuario administrador accederá en el menú superior al apartado añadir nuevos portlets. 3. El usuario seleccionará uno de los portlets y lo añadirá.	2. El sistema mostrará los diferentes portlets disponibles, que han sido previamente instalados. 4. El sistema agregará y cargará el portlet en el portal en una posición válida.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
POSTCONDICIÓN	El nuevo portlet se mantendrá mientras el usuario administrador no lo elimine, aunque se cierre la sesión.	

Tabla 48. CU-023 Desplegar portlet administrador

CASO DE USO	Replegar portlet administrador	
IDENTIFICADOR	CU-024	
ACTORES	Administrador	
DESCRIPCIÓN	El administrador podrá replegar un portlet desplegado en el portal.	
PRECONDICIÓN	El usuario deberá estar autenticado como administrador y el portlet desplegado en el portal.	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. El usuario administrador pulsará sobre el botón de cerrar portlet que se encontrará en la parte superior izquierda de cada portlet particular.	2. El sistema replegará el portlet del portal.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
POSTCONDICIÓN	El portlet no aparecerá de nuevo hasta que el usuario administrador lo vuelva a agregar al portal, aunque se cierre la sesión.	

Tabla 49. CU-024 Replegar portlet administrador

CASO DE USO	Crear páginas personales	
IDENTIFICADOR	CU-025	
ACTORES	Usuario miembro	
DESCRIPCIÓN	Un usuario miembro podrá crear páginas personales donde será el encargado de gestionar tanto los portlets que desee que aparezcan como la posición de los mismos en la página y la distribución de columnas.	
PRECONDICIÓN	El usuario deberá estar autenticado como usuario miembro.	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. El usuario accederá a su espacio personal a través de la barra superior. 2. Usará la opción *añadir página* en la barra superior y escribirá el nombre de la nueva página.	3. El sistema creará la página con el nombre indicado.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
POSTCONDICIÓN	La nueva página se mantendrá mientras el usuario no la elimine, aunque se cierre la sesión.	

Tabla 50. CU-025 Crear páginas personales

CASO DE USO	Desplegar portlet miembro	
IDENTIFICADOR	CU-026	
ACTORES	Usuario miembro	
DESCRIPCIÓN	Un usuario miembro podrá añadir un nuevo portlet a una de sus páginas personalizadas.	
PRECONDICIÓN	El usuario deberá estar autenticado como usuario miembro.	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. El usuario miembro accederá en el menú superior al apartado añadir nuevos portlets. 3. El usuario seleccionará uno de los portlets y lo añadirá.	2. El sistema mostrará los diferentes portlets disponibles, que han sido previamente instalados. 4. El sistema agregará y cargará el portlet a la página personal seleccionada, en una posición válida.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
POSTCONDICIÓN	El nuevo portlet se mantendrá mientras el usuario miembro no lo elimine o elimine la página donde esté desplegado, aunque se cierre la sesión.	

Tabla 51. CU-026 Desplegar portlet miembro

CASO DE USO	Replegar portlet miembro	
IDENTIFICADOR	CU-027	
ACTORES	Usuario miembro	
DESCRIPCIÓN	El usuario miembro podrá replegar un portlet de una de sus páginas personales.	
PRECONDICIÓN	El usuario deberá estar autenticado como miembro y el portlet desplegado en una página.	
ESCENARIO DE ÉXITO	**ACTORES**	**SISTEMA**
	1. El usuario miembro pulsará sobre el botón de cerrar portlet que se encontrará en la parte superior izquierda de cada portlet particular.	2. El sistema eliminará el portlet de la página del usuario.
ESCENARIO ALTERNATIVO	**ACTORES**	**SISTEMA**
POSTCONDICIÓN	El portlet no aparecerá de nuevo hasta que el usuario miembro lo vuelva a agregar a una de sus páginas, aunque se cierre la sesión.	

Tabla 52. CU-027 Replegar portlet miembro

Matriz de trazabilidad Casos de uso – Requisitos de software

A continuación se disponen las matrices de trazabilidad que muestran la relación entre los casos de uso y los requisitos de software descritos en el documento. Para una mayor claridad se han dividido los requisitos en los diferentes tipos ya comentados.

		Casos de Uso																										
		CU-001	CU-002	CU-003	CU-004	CU-005	CU-006	CU-007	CU-008	CU-009	CU-010	CU-011	CU-012	CU-013	CU-014	CU-015	CU-016	CU-017	CU-018	CU-019	CU-020	CU-021	CU-022	CU-023	CU-024	CU-025	CU-026	CU-027
Requisitos de software	RF-001	X																										
	RF-002		X																									
	RF-003		X																									
	RF-004		X																									
	RF-005																						X			X	X	X
	RF-006																						X					
	RF-007																					X						
	RF-008			X																								
	RF-009			X																								
	RF-010					X																						
	RF-011				X																							
	RF-012						X																					
	RF-013							X																				
	RF-014										X																	
	RF-015								X																			
	RF-016									X																		
	RF-017											X						X										
	RF-018													X														
	RF-019														X	X												
	RF-020												X															
	RF-021													X														
	RF-022																X											
	RF-023																		X									
	RF-024																		X	X	X							
	RF-025																			X								
	RF-026																			X								
	RF-027																			X								
	RF-028																				X							
	RF-029																				X							
	RF-030																				X							

Tabla 53. Matriz de trazabilidad. Requisitos funcionales-Casos de uso.

		Casos de Uso																											
		CU-001	CU-002	CU-003	CU-004	CU-005	CU-006	CU-007	CU-008	CU-009	CU-010	CU-011	CU-012	CU-013	CU-014	CU-015	CU-016	CU-017	CU-018	CU-019	CU-020	CU-021	CU-022	CU-023	CU-024	CU-025	CU-026	CU-027
Requisitos de software	RNF-RU-001			X						X																		
	RNF-RU-002			X			X			X				X														
	RNF-RU-003			X									X															
	RNF-RU-004			X																								
	RNF-RU-005						X																					
	RNF-RU-006							X																				
	RNF-RU-007													X														
	RNF-RU-008							X	X																			
	RNF-RU-009							X																				
	RNF-RU-010											X	X	X	X	X												

Tabla 54. Matriz de trazabilidad. Requisitos de usabilidad - Casos de uso.

		Casos de Uso																											
		CU-001	CU-002	CU-003	CU-004	CU-005	CU-006	CU-007	CU-008	CU-009	CU-010	CU-011	CU-012	CU-013	CU-014	CU-015	CU-016	CU-017	CU-018	CU-019	CU-020	CU-021	CU-022	CU-023	CU-024	CU-025	CU-026	CU-027
Requisitos de software	RNF-RI-001		X																									
	RNF-RI-002	X																										
	RNF-RI-003			X	X	X	X																					
	RNF-RI-004			X			X																					
	RNF-RI-005							X	X	X	X																	
	RNF-RI-006												X															
	RNF-RI-007													X														
	RNF-RI-008														X													
	RNF-RI-009															X												
	RNF-RI-010																X											
	RNF-RI-011																		X	X	X							
	RNF-RI-012																		X									
	RNF-RI-013																			X								
	RNF-RI-014																			X								
	RNF-RI-015																			X								
	RNF-RI-016																			X								
	RNF-RI-017																				X							
	RNF-RI-018																				X							

Tabla 55. Matriz de trazabilidad. Requisitos de interfaz - Casos de uso.

Apéndice B: Documento de requisitos de software

Introducción

Propósito

El objetivo de este documento es recoger el conjunto de requisitos a cumplir en el desarrollo del portal de sindicación de información deportiva y describir cada uno de ellos de forma que sean fáciles de comprender. También permitirá verificar que el producto cumple con su objetivo final.

Resumen

En este documento se encontrará la descripción general del producto a desarrollar junto con la lista de requisitos de software que el producto debe cumplir.

Los requisitos recogidos se describirán mediante tablas con diferentes atributos. Mediante estos atributos se conseguirá identificar de forma única cada requisito, se describirán de forma clara y concisa y se especificará la importancia y la necesidad de los mismos.

Los requisitos se dividirán en dos tipos principales, requisitos funcionales y no funcionales. Los requisitos no funcionales a su vez se dividirán en tres subcategorías, requisitos del sistema, de usabilidad y de interfaz.

Acrónimos

- **REST:** Representational State Transfer
- **RSS:** Really Simple Syndication
- **RF:** Requisito Funcional.
- **RNF:** Requisito no funcional.
- **RS:** Requisito de Sistema.

- **RU:** Requisito de Usabilidad.
- **RI:** Requisito de Interfaz.
- **IDE:** Integrated Development Environment.
- **API:** Application Programming Interface.
- **JDK:** Java Development kit.
- **JRE:** Java Runtime Environment
- **JSR:** Java Specification Request
- **WSDL:** Web Services Description Language

Definiciones

- **Slideshow:** Modo de presentación de noticias o imágenes que produce un desfile de estas de forma consecutiva y automática cada cierto tiempo.
- **Tweet:** Cada uno de los mensajes de texto escritos por los usuarios de Twitter. Están compuestos por 140 caracteres como máximo.
- **Feed RSS**: Fichero de tipo XML generado por los sitios web para ofrecer una versión reducida de la información incluida en el sitio web. Este archivo está compuesto por ítems que constan de un titular, un resumen de la información, un enlace a la noticia completa, autor, fecha de publicación y en algunas ocasiones, una fotografía que ilustra la información. El fichero se reescribe cuando se actualiza la información en el sitio web. De esta forma es posible acceder a la información del sitio web sin necesidad de visitarlo.

Descripción general

Perspectiva del producto

El producto a desarrollar consistirá en un portal de sindicación de información, que se obtendrá mediante diferentes tecnologías que permitirán obtenerla desde diversas fuentes web. El portal estará conformado por portlets. Cada portlet será desarrollado como un proyecto distinto con unos objetivos concretos dentro del portal web. Por lo tanto el proyecto estará formado por un conjunto de proyectos de menor tamaño que una vez unidos y desplegados en el portal ofrecerán la funcionalidad completa del producto a desarrollar.

Funcionalidad del producto

El producto será un portal sindicador de noticias deportivas. Su funcionalidad principal será ofrecer información deportiva actualizada sin la necesidad de redactores.

Esto es posible debido al uso de las diferentes tecnologías y técnicas objetivo de este proyecto, como son los servicios web, tanto los basados en SOAP como los basados en REST y la tecnología de distribución de contenidos RSS.

Mediante estas tecnologías y técnicas se conseguirá crear un portal de noticias actualizado, generado únicamente mediante la obtención de información ofrecida de forma gratuita por otros portales o sitios web con los mismos intereses que el portal que se va a desarrollar.

Requisitos de software

En este apartado se recogerán los requisitos de software del producto. Debido a que los portlets se pueden considerar como pequeños proyectos individuales, los requisitos de cada uno de ellos se expondrán en apartados diferenciados, aunque realmente sus identificadores sigan una numeración continua, de esta forma se entenderán más fácilmente y a su vez será más fácil realizar las matrices de trazabilidad. Los requisitos al mismo tiempo serán divididos en diferentes tipos según su propósito: Funcionales y no funcionales. Los no funcionales se dividirán en requisitos del sistema, de usabilidad y de interfaz.

Cada requisito será descrito mediante una tabla. Las tablas estarán compuestas por los siguientes atributos:

- *Identificador:* Es el nombre identificativo del requisito. Cada identificador será único y unívoco. Constará de una primera parte que indicará si el requisito es funcional (RF) o no funcional (RNF). En el caso de que el requisito sea no funcional, también constará con un segundo apartado que describirá que tipo de requisito no funcional es: Requisito de sistema (RS), requisito de usabilidad (RU), o requisito de interfaz (RI).
  Por último, tanto los requisitos funcionales como los no funcionales constarán de un número que definirá el orden del requisito dentro de cada grupo específico.
- *Nombre:* Nombre corto descriptivo del requisito.
- *Necesidad:* Definirá la necesidad del requisito dentro del proyecto. Existirán tres posibilidades: Esencial, que indicará que el requisito es muy importante, imprescindible en el objetivo del proyecto; Deseable, que corresponderá a los requisitos que son deseables pero no imprescindibles para el objetivo del proyecto y Opcional, para los requisitos que no influirían en el objetivo del proyecto si desaparecieran.
- *Prioridad:* Indicará la importancia de un requisito en comparación con el resto. Las posibilidades son alta, media y baja. Los requisitos con prioridad alta son los requisitos más importantes y serán elegidos para su implementación antes que los requisitos de menor prioridad, Los requisitos de baja prioridad serán los primeros abandonar en caso de falta de tiempo.

- *Descripción:* Explicación textual del requisito que será clara y concisa para que al leerla se pueda comprender el requisito fácil y rápidamente.

El esquema de una tabla de un requisito es el siguiente:

Identificador: Tipo-[subtipo]-XXX	
Nombre:	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción:	

Tabla 56. Esquema requisito.

Portal

Requisitos funcionales

Identificador: RF-001	
Nombre: Registrarse en el portal	
Necesidad: ☐ ESENCIAL ☐ DESEABLE ☑ OPCIONAL	**Prioridad: ☐ ALTA ☐ MEDIA ☑ BAJA**
Descripción: **Los visitantes del portal podrán registrarse en el portal si lo desean, convirtiéndose en usuarios del mismo.**	

Tabla 57. RF-001 Registrarse en el portal.

Identificador: RF-002	
Nombre: Autenticarse en el portal	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad: ☐ ALTA ☑ MEDIA ☐ BAJA**
Descripción: **Los usuarios del portal previamente registrados podrán acceder al mismo mediante un formulario de autenticación. Esto le permitirá realizar las opciones propias de su rol.**	

Tabla 58. RF-002 Autenticarse en el portal.

Identificador: RF-003	
Nombre: Recordar identificador y contraseña	
Necesidad: ☐ Esencial ☐ Deseable ☑ Opcional	**Prioridad:** ☐ Alta ☐ Media ☑ Baja
Descripción: El usuario que se va a autenticar podrá marcar una opción para que se mantengan los datos de su autenticación (identificador y contraseña). De esta manera, la próxima vez que el mismo usuario vaya a autenticarse en el portal no será necesario reintroducirlos.	

Tabla 59. RF-003 Recordar identificador y contraseña.

Identificador: RF-004	
Nombre: Desconexión del portal	
Necesidad: ☐ Esencial ☐ Deseable ☑ Opcional	**Prioridad:** ☐ Alta ☐ Media ☑ Baja
Descripción: Un usuario que se encuentre autenticado en el portal podrá pulsar un enlace que le desconectará del portal.	

Tabla 60. RF-004 Desconexión del portal.

Identificador: RF-005	
Nombre: Gestionar espacio personal	
Necesidad: ☐ Esencial ☑ Deseable ☐ Opcional	**Prioridad:** ☐ Alta ☑ Media ☐ Baja
Descripción: El usuario podrá personalizar su espacio personal mediante la barra de administración que aparecerá en la parte superior del portal una vez autenticado. Podrá crear nuevas páginas personales, añadir y eliminar nuevos portlets en sus propias páginas y colocarlos en la disposición que desee.	

Tabla 61. RF-005 Gestionar espacio personal.

Identificador: RF-006	
Nombre: Modificar disposición página principal	
Necesidad: ☐ Esencial ☑ Deseable ☐ Opcional	**Prioridad:** ☐ Alta ☑ Media ☐ Baja
Descripción: El usuario con derechos de administración podrá modificar la disposición de los portlets de la página principal mediante la barra de administración que aparecerá en la parte superior del portal una vez autenticado. Podrá desplegar nuevos portlets y replegar los ya desplegados. Todos los usuarios del portal verán esta disposición cuando accedan al portal a través del navegador.	

Tabla 62. RF-006 Modificar disposición página principal.

Identificador: RF-007	
Nombre: Modificar el esquema de columnas del portal	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: El usuario con derechos de administración podrá modificar el número y la disposición de las columnas del portal mediante la barra de administración que aparecerá en la parte superior del portal una vez autenticado. Todos los usuarios del portal verán esta disposición de columnas cuando accedan al portal a través del navegador.	

Tabla 63. RF-007 Modificar el esquema de columnas del portal.

Requisitos no funcionales

Requisitos del sistema

Identificador: RNF-RS-001	
Nombre: Compatibilidad con navegadores web	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: La aplicación deberá ser compatible con Internet Explorer 7.0 y superior y Mozilla Firefox 7.0 y superior.	

Tabla 64. RNF-RS-001 Compatibilidad con navegadores web.

Identificador: RNF-RS-002	
Nombre: Bases de datos	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: La aplicación utilizará MySQL 5.6 como sistema gestor de bases de datos.	

Tabla 65. RNF-RS-002 Bases de datos.

Identificador: RNF-RS-003	
Nombre: Portal gestor de contenidos	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: El portal gestor de los diferentes portlets será Liferay portal 6.0.5.	

Tabla 66. RNF-RS-003 Portal gestor de contenidos.

Identificador: RNF-RS-004	
Nombre: Servidor Web	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: El servidor web sobre el que se ejecutará la aplicación será GlassFish 3.0.1	

Tabla 67. RNF-RS-004 Servidor Web.

Identificador: RNF-RS-005	
Nombre: Especificación portlets	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: La versión a utilizar de la especificación de los portlets Java será la JSR 286: Portlet Specification 2.0.	

Tabla 68. RNF-RS-005 Especificación portlets.

Identificador: RNF-RS-006	
Nombre: IDE	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: El desarrollo de la aplicación se realizará con la ayuda del IDE NetBeans 6.9.1.	

Tabla 69. RNF-RS-006 IDE.

Identificador: RNF-RS-007	
Nombre: Lenguaje de desarrollo del lado del servidor	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: El código del lado del servidor será desarrollado en Java (JSP2.2 , Servlet3.0). Se utilizará el jdk 1.6.0 que incluye el jre6 para la ejecución del código java.	

Tabla 70. RNF-RS-007 Lenguaje de desarrollo del lado del servidor.

Identificador: RNF-RS-008	
Nombre: Librerías de Interfaz	
Necesidad: ☑ **ESENCIAL** ☐ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: **Para desarrollar la interfaz de usuario, se utilizarán la librerías JQuery v1.5.1 y JQuery UI v1.8.12.**	

Tabla 71. RNF-RS-008 Librerías de Interfaz.

Requisitos del interfaz

Identificador: RNF-RI-001	
Nombre: Formulario de autenticación	
Necesidad: ☐ **ESENCIAL** ☑ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **El portlet de autenticación estará formado por un campo de correo electrónico que funcionará como identificador y un campo contraseña. Además dispondrá de un checkbox para seleccionar si recordar los datos del autenticación. También aparecerá un enlace que conducirá al formulario de registro y otro para recuperar contraseña.**	

Tabla 72. RNF-RI-001 Formulario autenticación.

Identificador: RNF-RI-002	
Nombre: Formulario de registro	
Necesidad: ☐ **ESENCIAL** ☑ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **El formulario de registro estará compuesto por los campos nombre, segundo nombre, apellido, nombre de usuario, dirección de correo(ID), fecha de nacimiento, género y texto de verificación.**	

Tabla 73. RNF-RI-002 Formulario de registro.

Lector de tweets

Requisitos funcionales

Identificador: RF-008	
Nombre: Visualizar información vía servicios REST	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	**Prioridad:** ☑ Alta ☐ Media ☐ Baja
Descripción: El usuario podrá visualizar los mensajes escritos en Twitter por los equipos de fútbol de primera división y principales medios deportivos, obteniendo la información de un servicio web basado en REST.	

Tabla 74. RF-008 Visualizar información vía servicios REST.

Identificador: RF-009	
Nombre: Visualizar tweets en general	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	**Prioridad:** ☑ Alta ☐ Media ☐ Baja
Descripción: El usuario podrá ver los tweets de todos los usuarios de Twitter que estén siendo seguidos por la cuenta de Twitter de seguimiento. Esta cuenta consistirá en una cuenta normal de Twitter que seguirá a los usuarios de los equipos de la liga de fútbol de primera división y de los principales medios de comunicación deportivos. Mediante las APIS de Twitter definidas en el requisito RNF-RS-010, se obtendrá la información que se desea mostrar.	

Tabla 75. RF-009 Visualizar tweets en general.

Identificador: RF-010	
Nombre: Filtrar tweets por usuario	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	**Prioridad:** ☑ Alta ☐ Media ☐ Baja
Descripción: El usuario podrá visualizar los tweets filtrados por usuario para facilitar la visualización de los mensajes de un usuario concreto.	

Tabla 76. RF-010 Filtrar tweets por usuario.

Identificador: RF-011	
Nombre: Visualizar tweets antiguos	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	Prioridad: ☑ Alta ☐ Media ☐ Baja
Descripción: Los tweets mostrados se dividirán en páginas de diez mensajes. Los tweets más nuevos serán los mostrados en la página principal (la página 1). El usuario podrá seleccionar la página que desee visualizar para ver los tweets más antiguos.	

Tabla 77. RF-011 Visualizar tweets antiguos.

Identificador: RF-012	
Nombre: Visualizar tweet con foto	
Necesidad: ☐ Esencial ☐ Deseable ☑ Opcional	Prioridad: ☐ Alta ☐ Media ☑ Baja
Descripción: Los tweets que dispongan de fotografía podrán ser visualizados individualmente de forma que la imagen se muestre a mayor tamaño junto con el resto de información.	

Tabla 78. RF-012 Visualizar tweets con foto.

Requisitos no funcionales

Requisitos del sistema

Identificador: RNF-RS-009	
Nombre: Uso tecnología AJAX	
Necesidad: ☐ Esencial ☑ Deseable ☐ Opcional	Prioridad: ☐ Alta ☐ Media ☑ Baja
Descripción: Para la elección de usuario y página a mostrar en el portlet de Twitter y para la actualización automática del tablón de tweets, se utilizará la función POST de JQuery para realizar llamadas asíncronas al servidor y evitar la actualización completa del portal.	

Tabla 79. RNF-RS-009 Uso tecnología AJAX.

Identificador: RNF-RS-010	
Nombre: Uso de las APIs REST y Stream de Twitter	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	Prioridad: ☑ Alta ☐ Media ☐ Baja
Descripción: Para obtener la información de Twitter se usarán las APIs ofrecidas por el mismo. La API REST de Twitter permite obtener la información mediante una simple interfaz. El API Stream de Twitter permite obtener información en tiempo real.	

Tabla 80. RNF-RS-010 Uso de las APIs REST y Stream de Twitter.

Identificador: RNF-RS-011	
Nombre: Uso de la librería Twitter4j	
Necesidad: ☐ **ESENCIAL** ☑ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **Se utilizará la versión 3.0.3 de la librería Twitter4j para manejar las APIs REST y Stream de Twitter en la aplicación.**	

Tabla 81. RNF-RS-011 Uso de la librería Twitter4j.

Identificador: RNF-RS-012	
Nombre: Almacenamiento de tweets en base de datos	
Necesidad: ☑ **ESENCIAL** ☐ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: **El sistema almacenará los mensajes en una base de datos para poder visualizarlos por los diferentes criterios (General o por usuario y antigüedad).**	

Tabla 82. RNF-RS-012 Almacenamiento de tweets en base de datos.

Identificador: RNF-RS-013	
Nombre: Actualización automática portlet REST	
Necesidad: ☑ **ESENCIAL** ☐ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: **El sistema actualizará el portlet que visualiza tweets para mantenerlo actualizado en todo momento (cada 15 segundos).**	

Tabla 83. RNF-RS-013 Actualización automática portlet REST.

Requisitos de usabilidad

Identificador: RNF-RU-001	
Nombre: Abrir los Hipervínculos en pestañas diferentes	
Necesidad: ☐ **ESENCIAL** ☑ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **Los hipervínculos a páginas externas se abrirán en pestañas diferentes para que los usuarios no pierdan el contexto del portal. *(Patrón K8 External links).***	

Tabla 84. RNF-RU-001 Abrir los Hipervínculos en pestañas diferentes.

Identificador: RNF-RU-002	
Nombre: Hipervínculos fáciles de identificar	
Necesidad: ☐ **ESENCIAL** ☑ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **Todos los hipervínculos deben ser fáciles de identificar. Deberán diferenciarse de forma clara cuando un hipervínculo ha sido visitado y cuando no cambiando el color. Los hipervínculos estarán subrayados y también cambiarán de color cuando se pase el ratón por encima.** ***(Patrón K10 Obvious links).***	

Tabla 85. RNF-RU-002 Hipervínculos fáciles de identificar.

Identificador: RNF-RU-003	
Nombre: Ajuste de maquetación automática	
Necesidad: ☐ **ESENCIAL** ☑ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **La maquetación del lector de tweets cambiará en tiempo de ejecución para ajustarse y optimizar su visualización, teniendo en cuenta el tamaño de la columna donde se localice.**	

Tabla 86. RNF-RU-003 Ajuste de maquetación automática.

Identificador: RNF-RU-004	
Nombre: Estilo de tweets	
Necesidad: ☐ **ESENCIAL** ☑ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **La información de los tweets que se muestren tendrá un formato y color similar al que usa Twitter para que sea fácil diferenciar cada parte del mensaje, haciendo estos reconocibles para los usuarios del programa.**	

Tabla 87. RNF-RU-004 Estilo de tweets.

Identificador: RNF-RU-005	
Nombre: Ventana modal para tweets con foto	
Necesidad: ☐ **ESENCIAL** ☑ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **Los tweets que dispongan de imagen se podrán expandir para ver esta a un tamaño mayor junto con el resto de información. Para hacer esto se hará uso de una ventana modal que permitirá al usuario volver al contexto anterior simplemente cerrando esta ventana.** ***(Patrón H6 Helping users complete tasks).***	

Tabla 88. RNF-RU-005 Ventana modal para Tweets con foto.

Requisitos de interfaz

Identificador: RNF-RI-003	
Nombre: Disposición portlet de Twitter vía REST	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: El portlet de mensajes de Twitter tendrá un tablón central donde se mostrarán los tweets. En la parte superior del tablón habrá un menú desplegable con el que se seleccionará el usuario del cual se desea ver los mensajes. En la parte inferior del tablón habrá otro menú desplegable con el que se seleccionará la página de tweets a mostrar, siendo la número uno la de los tweets más modernos.	

Tabla 89. RNF-RI-003 Disposición portlet de Twitter vía REST.

Identificador: RNF-RI-004	
Nombre: Información de Tweet	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: De cada tweet se mostrará la imagen de usuario, el nombre de usuario, el texto y la fecha.	

Tabla 90. RNF-RI-004 Información de Tweet.

Lector RSS

Requisitos funcionales

Identificador: RF-013	
Nombre: Visualizar noticias vía RSS	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: El usuario podrá visualizar noticias deportivas obtenidas a través de los canales RSS de los principales medios de información.	

Tabla 91. RF-013 Visualizar noticias vía RSS.

Identificador: RF-014	
Nombre: Seleccionar feed RSS a mostrar	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	Prioridad: ☑ Alta ☐ Media ☐ Baja
Descripción: El usuario podrá seleccionar el canal RSS del medio de información que desee visualizar. Una vez seleccionado, el portlet cargará las últimas noticias introducidas en ese canal y las mostrará.	

Tabla 92. RF-014 Seleccionar feed RSS a mostrar .

Identificador: RF-015	
Nombre: Seleccionar noticia a mostrar	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	Prioridad: ☑ Alta ☐ Media ☐ Baja
Descripción: El usuario, mediante el ratón, podrá seleccionar la noticia sobre la que se hará foco de la lista de noticias mostradas. Las noticias se mostrarán en forma de slideshow, que irá cambiando la noticia mostrada cada cierto tiempo. Cuando se realice el foco sobre una noticia el giro automático se detendrá hasta que se retire el ratón.	

Tabla 93. RF-015 Seleccionar noticia a mostrar.

Identificador: RF-016	
Nombre: Acceder a la noticia completa	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	Prioridad: ☑ Alta ☐ Media ☐ Baja
Descripción: El usuario podrá acceder a la noticia completa mediante un hipervínculo en su titular, siendo redireccionado a la página de la noticia en el sitio web de la fuente.	

Tabla 94. RF-016 Acceder a la noticia completa.

Requisitos no funcionales

Requisitos del sistema

Identificador: RNF-RS-014	
Nombre: Uso del API Google Feed	
Necesidad: ☐ Esencial ☑ Deseable ☐ Opcional	Prioridad: ☐ Alta ☑ Media ☐ Baja
Descripción: Para la lectura de los feeds RSS a partir de su URL, se utilizará el API Google Feed v1.	

Tabla 95. RNF-RS-014 Uso del API Google Feed.

Identificador: RNF-RS-015	
Nombre: Actualización automática portlet RSS	
Necesidad: ☑ **ESENCIAL** ☐ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: **El sistema actualizará automáticamente las noticias mostradas en el portlet RSS a los más recientes cada 15 minutos.**	

Tabla 96. RNF-RS-015 Actualización automática portlet RSS.

Identificador: RNF-RS-016	
Nombre: Uso de JQuery	
Necesidad: ☐ **ESENCIAL** ☑ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **Para el desarrollo de JavaScript y la creación de la interfaz se utilizará la librería JQuery v1.5.1 y JQuery UI v1.8.12.**	

Tabla 97. RNF-RS-016 Uso de JQuery.

Requisitos de usabilidad

Identificador: RNF-RU-001	
Nombre: Abrir los Hipervínculos en pestañas diferentes	
Necesidad: ☐ **ESENCIAL** ☑ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **Los hipervínculos a páginas externas se abrirán en pestañas diferentes para que los usuarios no pierdan el contexto del portal. *(Patrón K8 External links).***	

Tabla 98. RNF-RU-001 Abrir los Hipervínculos en pestañas diferentes.

Identificador: RNF-RU-006	
Nombre: Forma corta de las noticias RSS	
Necesidad: ☐ **ESENCIAL** ☑ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **En el portlet de noticias obtenidas mediante RSS se mostrará el titular junto una imagen representativa si es ofrecida por el feed y un resumen de la noticia. En el supuesto que el usuario desee leer la noticia completa, se accederá al sitio web de la fuente mediante un hipervínculo. *(Patrón A2 News mosaics)*.**	

Tabla 99. RNF-RU-006 Forma corta de las noticias RSS.

Identificador: RNF-RU-007	
Nombre: Usar nombres descriptivos en los hipervínculos	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad: ☐ ALTA ☑ MEDIA ☐ BAJA**
Descripción: **Los nombres utilizados en los hipervínculos serán los titulares de la noticias. De esta manera serán intuitivos. *(Patrón K9 Descriptive, Longer Link Names)*.**	

Tabla 100. RNF-RU-007 Usar nombres descriptivos en los hipervínculos.

Identificador: RNF-RU-008	
Nombre: Slideshow para los Feeds	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad: ☐ ALTA ☑ MEDIA ☐ BAJA**
Descripción: **El sistema mostrará las noticias más recientes de los feeds mediante un slideshow que cambiará la noticia a mostrar de forma automática cada 7 segundos en un bucle circular.**	

Tabla 101. RNF-RU-008 Slideshow para los Feeds.

Identificador: RNF-RU-009	
Nombre: Ajuste de la visualización Feeds	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad: ☐ ALTA ☑ MEDIA ☐ BAJA**
Descripción: **El sistema analizará el tamaño de la columna donde se encuentre el portlet RSS y adecuará su disposición a este tamaño para facilitar su visualización.**	

Tabla 102. RNF-RU-009 Ajuste de la visualización Feeds.

Requisitos de interfaz

Identificador: RNF-RI-005	
Nombre: Disposición portlet de noticias vía RSS	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad: ☐ ALTA ☑ MEDIA ☐ BAJA**
Descripción: **El portlet de noticias que muestre feeds RSS se compondrá de dos zonas: La parte derecha contendrá los titulares de las últimas noticias. La parte izquierda tendrá tres subapartados, el titular de la noticia, la fotografía y un pequeño resumen. Cuando se pase el ratón por encima de un titular de la zona derecha aparecerá su titular, fotografía y resumen correspondiente en la parte izquierda. Además dispondrá de un menú desplegable para seleccionar el feed RSS del medio informativo que se desee entre los ofrecidos.**	

Tabla 103. RNF-RI-005 Disposición portlet de noticias vía RSS.

Información general del Mundial 2014

Requisitos funcionales

Identificador: RF-017	
Nombre: Visualizar información vía servicios web basados en SOAP	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	**Prioridad:** ☑ Alta ☐ Media ☐ Baja
Descripción: El usuario podrá visualizar información sobre el Mundial de Brasil 2014, obtenida a través de un servicio web basado en SOAP. A partir de un fichero WSDL se obtendrá toda la información.	

Tabla 104. RF-017 Visualizar información vía servicios web.

Identificador: RF-018	
Nombre: Visualizar plantilla selecciones Mundial 2014	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	**Prioridad:** ☑ Alta ☐ Media ☐ Baja
Descripción: El usuario podrá visualizar las plantillas de cada selección del Mundial de fútbol 2014 divididas por posiciones de los jugadores. Se accederá haciendo clic sobre cualquiera de los equipos de la página de grupos.	

Tabla 105. RF-018 Visualizar plantilla selecciones Mundial 2014.

Identificador: RF-019	
Nombre: Visualizar estadios Mundial 2014	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	**Prioridad:** ☑ Alta ☐ Media ☐ Baja
Descripción: El usuario podrá visualizar la información sobre los estadios del Mundial 2014. Podrá seleccionar el estadio sobre el que desee informarse a partir de la tabla de estadios.	

Tabla 106. RF-019 Visualizar estadios Mundial 2014.

Identificador: RF-020	
Nombre: Visualizar grupos Mundial 2014	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	**Prioridad:** ☑ Alta ☐ Media ☐ Baja
Descripción: El usuario podrá visualizar la información sobre los grupos del Mundial 2014.	

Tabla 107. RF-020 Visualizar grupos Mundial 2014.

Identificador: RF-021	
Nombre: Volver a página principal	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: El usuario podrá volver a la página principal del portlet (página de grupos) mediante un botón.	

Tabla 108. RF-021 Volver a página principal.

Requisitos no funcionales

Requisitos del sistema

Identificador: RNF-RS-017	
Nombre: Uso EJB	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: Como varios portlets van a acceder al mismo servicio web, se utilizara un EJB como cliente del servicio, de esta forma se evitará tener que crear un cliente por cada portlet, así como la repetición de las clases generadas por el cliente en cada uno de ellos.	

Tabla 109. RNF-RS-017 Uso EJB.

Requisitos de usabilidad

Identificador: RNF-RU-002	
Nombre: Hipervínculos fáciles de identificar	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: Todos los hipervínculos deben ser fáciles de identificar. Deberán diferenciarse de forma clara cuando un hipervínculo ha sido visitado y cuando no cambiando el color. Los hipervínculos estarán subrayados y también cambiarán de color cuando se pase el ratón por encima. ***(Patrón K10 Obvious links).***	

Tabla 110. RNF-RU-002 Hipervínculos fáciles de identificar.

Identificador: RNF-RU-010	
Nombre: Camino de navegación	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **Se mostrará el camino recorrido durante la navegación en este portlet para saber en qué lugar se encuentra el usuario en cada momento. *(Patrón K6 Bread Crumbs).***	

Tabla 111. RNF-RU-010 Camino de navegación.

Requisitos de interfaz

Identificador: RNF-RI-006	
Nombre: Acceso a las plantillas de las selecciones del Mundial 2014	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **Para el acceso a las plantillas de cada selección nacional, el portlet dispondrá los grupos oficiales del Mundial 2014 conteniendo las selecciones correspondientes en cada uno. El nombre de cada selección será un hipervínculo que conducirá hasta su plantilla.**	

Tabla 112. RNF-RI-006 Acceso a las plantillas de las selecciones del Mundial 2014.

Identificador: RNF-RI-007	
Nombre: Plantillas selecciones nacionales	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **Para visualizar las plantillas completas de las selecciones nacionales se dispondrá en un tablón a los jugadores divididos por su posición en el campo. Para regresar a la pantalla de elegir selección nacional existirá un enlace "volver" o se podrá regresar también mediante el camino de navegación.**	

Tabla 113. RNF-RI-007 Plantillas selecciones nacionales.

Identificador: RNF-RI-008	
Nombre: Acceso a los estadios del Mundial 2014	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **Para el acceso a la información de cada estadio, el portlet dispondrá de un listado de los estadios con su foto. Cada foto será un hipervínculo a toda la información sobre ese estadio.**	

Tabla 114. RNF-RI-008 Acceso a los estadios del Mundial 2014.

Identificador: RNF-RI-009	
Nombre: Información de los estadios del Mundial 2014	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: La información de los estadios estará compuesta por el nombre del estadio, la ciudad, la capacidad del estadio, el enlace a la página web sobre el estadio en wikipedia, la imagen del estadio y el mapa de Google Maps con la situación del estadio. Para regresar a la pantalla de elección de estadio existirá un enlace "volver" o se podrá regresar también mediante el camino de navegación.	

Tabla 115. RNF-RI-009 Información de los estadios del Mundial 2014.

Clasificación goleadores Mundial 2014

Requisitos funcionales

Identificador: RF-022	
Nombre: Visualizar goleadores Mundial 2014	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: El usuario podrá visualizar los máximos goleadores del Mundial 2014	

Tabla 116. RF-022 Visualizar goleadores Mundial 2014.

Requisitos no funcionales

Requisitos del sistema

Identificador: RNF-RS-017	
Nombre: Uso EJB	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: Como varios portlets van a acceder al mismo servicio web, se utilizara un EJB como cliente del servicio, de esta forma se evitará tener que crear un cliente por cada portlet, así como la repetición de las clases generadas por el cliente en cada uno de ellos.	

Tabla 117. RNF-RS-017 Uso EJB.

Requisitos de interfaz

Identificador: RNF-RI-010	
Nombre: Información de los goleadores del Mundial 2014	
Necesidad: ☐ Esencial ☑ Deseable ☐ Opcional	Prioridad: ☐ Alta ☑ Media ☐ Baja
Descripción: La Información de los goleadores constará de un listado de los diez máximos goleadores del Mundial 2014. De cada goleador se mostrará el nombre, la bandera de su nacionalidad y el número de goles conseguidos.	

Tabla 118. RNF-RI-010 Información de los goleadores del Mundial 2014.

Clasificación Liga LFP

Requisitos funcionales

Identificador: RF-017	
Nombre: Visualizar información vía servicios web basados en SOAP	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	Prioridad: ☑ Alta ☐ Media ☐ Baja
Descripción: El usuario podrá visualizar información sobre la LFP, obtenida a través de un servicio web basado en SOAP. A partir de un fichero WSDL se obtendrá toda la información.	

Tabla 119. RF-017 Visualizar información vía servicios web.

Identificador: RF-023	
Nombre: Visualizar clasificación primera división	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	Prioridad: ☑ Alta ☐ Media ☐ Baja
Descripción: El usuario podrá visualizar la información correspondiente a la clasificación de los equipos de la liga LFP de primera división.	

Tabla 120. RF-023 Visualizar clasificación primera división.

Identificador: RF-024	
Nombre: Ordenación de tabla	
Necesidad: ☐ ESENCIAL ☐ DESEABLE ☑ OPCIONAL	**Prioridad:** ☐ ALTA ☐ MEDIA ☑ BAJA
Descripción: El usuario podrá ordenar la información de la tabla por cada una de las columnas de esta para ver la información ordenada descendente o ascendentemente.	

Tabla 121. RF-024 Ordenación de tabla.

Requisitos no funcionales

Requisitos del sistema

Identificador: RNF-RS-016	
Nombre: Uso de JQuery	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: Para el desarrollo de JavaScript y la creación de la interfaz se utilizará la librería JQuery v1.5.1 y JQuery UI v1.8.12	

Tabla 122. RNF-RS-016 Uso de JQuery.

Identificador: RNF-RS-017	
Nombre: Uso EJB	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: Como varios portlets van a acceder al mismo servicio web, se utilizara un EJB como cliente del servicio, de esta forma se evitará tener que crear un cliente por cada portlet, así como la repetición de las clases generadas por el cliente en cada uno de ellos.	

Tabla 123. RNF-RS-017 Uso EJB.

Requisitos de interfaz

Identificador: RNF-RI-011	
Nombre: Tabla ordenable	
Necesidad: ☐ ESENCIAL ☐ DESEABLE ☑ OPCIONAL	**Prioridad:** ☐ ALTA ☐ MEDIA ☑ BAJA
Descripción: Se deberá implementar una tabla ordenable que permita ordenar su contenido ascendente y descendentemente dependiendo de cada columna.	

Tabla 124. RNF-RI-011 Tabla ordenable.

Identificador: RNF-RI-012	
Nombre: Información tabla clasificación LFP	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: La tabla de clasificación de la liga LFP deberá mostrar la posición de cada equipo con un número, una imagen que señale si el equipo está en Europa o en puestos de descenso, el nombre del club, una imagen del escudo, los partidos jugados, los partidos ganados, los partidos empatados, los partidos perdidos, los goles a favor, los goles en contra y la diferencia de goles de cada equipo.	

Tabla 125. RNF-RI-012 Información tabla clasificación LFP.

Estadísticas Liga LFP

Requisitos funcionales

Identificador: RF-017	
Nombre: Visualizar información vía servicios web basados en SOAP	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: El usuario podrá visualizar información sobre la LFP, obtenida a través de un servicio web basado en SOAP. A partir de un fichero WSDL se obtendrá toda la información.	

Tabla 126. RF-017 Visualizar información vía servicios web.

Identificador: RF-024	
Nombre: Ordenación de tabla	
Necesidad: ☐ **ESENCIAL** ☐ **DESEABLE** ☑ **OPCIONAL**	**Prioridad:** ☐ ALTA ☐ MEDIA ☑ BAJA
Descripción: **El usuario podrá ordenar la información de la tabla por cada una de las columnas de esta para ver la información ordenada descendente o ascendentemente.**	

Tabla 127. RF-024 Ordenación de tabla.

Identificador: RF-025	
Nombre: Visualizar clasificación de goleadores	
Necesidad: ☑ **ESENCIAL** ☐ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: **El usuario podrá ver la información de goleadores de la liga LFP.**	

Tabla 128. RF-025 Visualizar clasificación de goleadores.

Identificador: RF-026	
Nombre: Visualizar clasificación de asistencias	
Necesidad: ☑ **ESENCIAL** ☐ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: **El usuario podrá ver la información de asistentes de la liga LFP.**	

Tabla 129. RF-026 Visualizar clasificación de asistencias.

Identificador: RF-027	
Nombre: Visualizar clasificación de amonestaciones	
Necesidad: ☑ **ESENCIAL** ☐ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: **El usuario podrá ver la información de amonestaciones de la liga LFP.**	

Tabla 130. RF-027 Visualizar clasificación de amonestaciones.

Requisitos no funcionales

Requisitos del sistema

Identificador: RNF-RS-016	
Nombre: Uso de JQuery	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: Para el desarrollo de JavaScript y la creación de la interfaz se utilizará la librería JQuery v1.5.1 y JQuery UI v1.8.12	

Tabla 131. RNF-RS-016 Uso de JQuery.

Identificador: RNF-RS-017	
Nombre: Uso EJB	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Descripción: Como varios portlets van a acceder al mismo servicio web, se utilizara un EJB como cliente del servicio, de esta forma se evitará tener que crear un cliente por cada portlet, así como la repetición de las clases generadas por el cliente en cada uno de ellos.	

Tabla 132. RNF-RS-017 Uso EJB.

Requisitos de interfaz

Identificador: RNF-RI-011	
Nombre: Tabla ordenable	
Necesidad: ☐ ESENCIAL ☐ DESEABLE ☑ OPCIONAL	**Prioridad:** ☐ ALTA ☐ MEDIA ☑ BAJA
Descripción: Se deberá implementar una tabla ordenable que permita ordenar su contenido ascendente y descendentemente dependiendo de cada columna.	

Tabla 133. RNF-RI-011 Tabla ordenable.

Identificador: RNF-RI-013	
Nombre: Disposición portlet estadísticas LFP	
Necesidad: ☐ **ESENCIAL** ☑ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **El portlet de estadísticas de la LFP se compondrá de un tablón central donde se mostrará la tabla correspondiente con la información. En la parte superior tendrá tres botones para poder cambiar la tabla que se quiera visualizar en el tablón, ya sea la clasificación de goleadores, la de asistencias o la de amonestaciones.**	

Tabla 134. RNF-RI-013 Disposición portlet estadísticas LFP.

Identificador: RNF-RI-014	
Nombre: Información tabla goleadores	
Necesidad: ☐ **ESENCIAL** ☑ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **La tabla de clasificación de goleadores deberá mostrar la foto, el nombre, el escudo del equipo al que corresponde y el número de goles de cada jugador.**	

Tabla 135. RNF-RI-014 Información tabla goleadores.

Identificador: RNF-RI-015	
Nombre: Información tabla asistencias	
Necesidad: ☐ **ESENCIAL** ☑ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **La tabla de clasificación de asistencias deberá mostrar la foto, el nombre, el escudo del equipo al que corresponde y el número de asistencias de cada jugador.**	

Tabla 136. RNF-RI-015 Información tabla asistencias.

Identificador: RNF-RI-016	
Nombre: Información tabla amonestaciones	
Necesidad: ☐ **ESENCIAL** ☑ **DESEABLE** ☐ **OPCIONAL**	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: **La tabla de clasificación de amonestaciones deberá mostrar la foto, el nombre, el escudo del equipo al que corresponde, el número de tarjetas rojas y el número de tarjetas amarillas de cada jugador.**	

Tabla 137. RNF-RI-016 Información tabla amonestaciones.

Plantillas Liga LFP

Requisitos funcionales

Identificador: RF-017	
Nombre: Visualizar información vía servicios web basados en SOAP	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	**Prioridad: ☑ Alta ☐ Media ☐ Baja**
Descripción: El usuario podrá visualizar información sobre la LFP, obtenida a través de un servicio web basado en SOAP. A partir de un fichero WSDL se obtendrá toda la información.	

Tabla 138. RF-017 Visualizar información vía servicios web.

Identificador: RF-024	
Nombre: Ordenación de tabla	
Necesidad: ☐ Esencial ☐ Deseable ☑ Opcional	**Prioridad: ☐ Alta ☐ Media ☑ Baja**
Descripción: El usuario podrá ordenar la información de la tabla por cada una de las columnas de esta para ver la información ordenada descendente o ascendentemente.	

Tabla 139. RF-024 Ordenación de tabla.

Identificador: RF-028	
Nombre: Visualizar equipos	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	**Prioridad: ☑ Alta ☐ Media ☐ Baja**
Descripción: El usuario podrá ver una tabla con los equipos de primera división.	

Tabla 140. RF-028 Visualizar equipos.

Identificador: RF-029	
Nombre: Visualizar plantilla de equipo	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	**Prioridad: ☑ Alta ☐ Media ☐ Baja**
Descripción: El usuario podrá ver la plantilla completa de un equipo seleccionado.	

Tabla 141. RF-029 Visualizar plantilla de equipo.

Identificador: RF-030	
Nombre: Volver a página de equipos	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	Prioridad: ☑ Alta ☐ Media ☐ Baja
Descripción: El usuario podrá ver volver a la página de equipos desde cualquier plantilla.	

Tabla 142. RF-030 Volver a página de equipos.

Requisitos no funcionales

Requisitos del sistema

Identificador: RNF-RS-016	
Nombre: Uso de JQuery	
Necesidad: ☐ Esencial ☑ Deseable ☐ Opcional	Prioridad: ☐ Alta ☑ Media ☐ Baja
Descripción: Para el desarrollo de JavaScript y la creación de la interfaz se utilizará la librería JQuery v1.5.1 y JQuery UI v1.8.12	

Tabla 143. RNF-RS-016 Uso de JQuery.

Identificador: RNF-RS-017	
Nombre: Uso EJB	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	Prioridad: ☑ Alta ☐ Media ☐ Baja
Descripción: Como varios portlets van a acceder al mismo servicio web, se utilizara un EJB como cliente del servicio, de esta forma se evitará tener que crear un cliente por cada portlet, así como la repetición de las clases generadas por el cliente en cada uno de ellos.	

Tabla 144. RNF-RS-017 Uso EJB.

Requisitos de interfaz

Identificador: RNF-RI-011	
Nombre: Tabla ordenable	
Necesidad: ☐ ESENCIAL ☐ DESEABLE ☑ OPCIONAL	**Prioridad:** ☐ ALTA ☐ MEDIA ☑ BAJA
Descripción: Se deberá implementar una tabla ordenable que permita ordenar su contenido ascendente y descendentemente dependiendo de cada columna.	

Tabla 145. RNF-RI-011 Tabla ordenable.

Identificador: RNF-RI-017	
Nombre: Disposición portlet plantilla LFP	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: El portlet de plantillas de los equipos de la LFP se compondrá de una pantalla inicial con los escudos de los equipos de primera división, sirviendo estos como links que permitirán acceder a la información de las plantillas de cada equipo. Esta información será presentada en una tabla ordenable. Desde cualquier pantalla de plantilla se podrá volver a la página principal mediante un botón de volver.	

Tabla 146. RNF-RI-017 Disposición portlet plantilla LFP.

Identificador: RNF-RI-018	
Nombre: Información tabla plantilla	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Descripción: La tabla de la plantilla de un equipo deberá mostrar el dorsal, la foto, el nombre, la posición, el número de goles, el número de asistencias, el número de tarjetas rojas y el número de tarjetas amarillas de cada jugador.	

Tabla 147. RNF-RI-018 Información tabla plantilla.